ELSEVIER

Theoretical Computer Science 266 (2001) 569–603

Theoretical Computer Science

www.elsevier.com/locate/tcs

Bisimulations in the join-calculus

Cédric Fournet[a], Cosimo Laneve[b,*,1]

[a] *Microsoft Research, 1 Guildhall Street, Cambridge, UK*

[b] *Department of Computer Science, Università di Bologna, Mura Anteo Zamboni 7, 40127 Bologna, Italy*

Received June 1999; revised May 2000; accepted June 2000

Communicated by U. Montanari

Abstract

We develop a theory of bisimulations in the join-calculus. We introduce a refined operational model that makes interactions with the environment explicit, and discuss the impact of the lexical scope discipline of the join-calculus on its extensional semantics. We propose several formulations of bisimulation and establish that all formulations yield the same equivalence. We prove that this equivalence is finer than barbed congruence, but that both relations coincide in the presence of name matching. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Asynchronous processes; Barbed congruence; Bisimulation; Chemical semantics; Concurrency; Join-calculus; Locality; Name matching; π-calculus

1. Introduction

The join-calculus is a recent formalism for modeling mobile systems [15, 17]. Its main motivation is to relate two crucial issues in concurrency: distributed implementation and formal semantics. To this end, the join-calculus enforces a strict lexical scope discipline over the channel names that appear in processes: names can be sent and received, but their input capabilities cannot be affected by the receivers. This is the *locality property*.[2]

Locality yields a realistic distributed model, because the communication primitives of the calculus can be directly implemented via standard primitives of asynchronous

* Corresponding author.

E-mail address: laneve@cs.unibo.it (C. Laneve).

[1] This work is partly supported by the ESPRIT CONFER-2 WG-21836.

[2] The term locality is a bit overloaded in the literature; here, names are locally defined inasmuch as no external definition may interfere; this is the original meaning of locality in the chemical semantics of Banâtre et al. [5] and in other papers on the join-calculus.

PII: S0304-3975(00)00283-8

systems [17, 19, 21]. It also plays a prominent role in the design of implicit type systems for the join-calculus, because all contravariant occurrences of names are static [18]. In this paper, we show that locality strongly affects the treatment of bisimulation and leads to simple semantics.

In order to reason about distributed processes, the join-calculus should be equipped with semantics that have both a sensible discriminating power and some convenient proof techniques. The usual approach in concurrency is to focus on elementary forms of interaction between the process and its environment; this can be achieved simply by defining a *reduction relation* that represents internal evolution and an *observation predicate* that detects the ability of interacting. Based on these two notions, numerous observational semantics can be defined. As regards discriminating power, it would be adequate to test the observation predicate under all possible contexts. In order to establish testing equivalences, however, one must cope with quantification over both contexts and series of reductions; this makes direct proofs particularly difficult [23]. To tackle these problems, it is convenient to consider equivalences that are finer and easier to check, possibly using them as indirect proof techniques for coarser semantics. Among them, bisimulation-based semantics are especially convenient, because co-inductive proofs need to consider only single reduction steps instead of traces [26]. *Barbed bisimilarity* has been proposed in [29] as a uniform basis to define behavioral equivalences on different process calculi. This is the approach taken so far for the join-calculus in [15, 17, 1], where *barbed congruence* is defined as the coarsest congruence that is a barbed bisimulation. Yet, checking that two processes are barbed congruent still requires explicit quantification over all contexts. This makes routine or automated checking of these properties problematic.

In this paper, we introduce a labeled operational model for the join calculus and we equip it with the standard *weak bisimulation* [26]. We illustrate the use of purely co-inductive proof techniques on labeled transitions. We also consider alternative definitions of bisimulation and relate weak bisimilarity to barbed congruence, showing that the two semantics coincide only in the presence of name matching. A more precise account of our work follows.

The original semantics of the join-calculus is based on the *reflexive chemical abstract machine* (RCHAM) [15]. This model accounts for the internal evolution of processes of the form $\mathsf{def}\ D\ \mathsf{in}\ P$ by describing how name definitions D can be used to receive and synchronize messages sent on these names in P. In order to adapt labeled bisimulation to the join-calculus, we propose a refined semantics – the *open* RCHAM – that makes explicit the interactions with the environment. Via these interactions, the environment can receive locally defined names of the process when they are emitted on free names, and the environment can also emit messages on these names. We call these interactions *extrusions* and *intrusions*, respectively. To keep track of the defined names that are visible from the environment, definitions of the join-calculus are marked with their extruded names when extrusions occur. In turn, intrusions are allowed only on names that are marked as extruded. The refined syntax for the join-calculus has processes of the form $\mathsf{def}_S\ D\ \mathsf{in}\ P$, where S is the set of names defined by D and extruded to the

environment. Informally, extruded names represent constants in the input interface of the process.

Let us illustrate locality in our setting: the process $\mathsf{def}_{\{x\}}\ x\langle u\rangle \triangleright P\ \mathsf{in}\ x\langle v\rangle$ defines a name x and sends a message $x\langle v\rangle$ on that name. According to the definition of x, a fresh copy of the process P is started whenever a message $x\langle u\rangle$ is received. Since the name x is marked as extruded, the environment can send messages on x to trigger copies of P. However, the environment cannot interfere with the definition of x; in particular, the message $x\langle v\rangle$ cannot be consumed by the environment, hence for any observational equivalence we should have the equation

$$\mathsf{def}_{\{x\}}\ x\langle u\rangle \triangleright P\ \mathsf{in}\ x\langle v\rangle = \mathsf{def}_{\{x\}}\ x\langle u\rangle \triangleright P\ \mathsf{in}\ P\{v/u\}.$$

Once the open RCHAM has been defined, weak bisimulation is obtained by applying standard definitions. The largest weak bisimulation, named *weak bisimilarity*, is preserved by renaming, is a congruence for all contexts of the open calculus, and suitably abstracts from the actual structure of the definitions. In particular, weak bisimilarity is insensitive to message buffering, as can be expected from an asynchronous semantics.

The open RCHAM allows the intrusion of messages on extruded names, regardless of the state of the process. This is crucial for hiding the presence of messages on defined names, but this also saturates bisimulations with processes that differ only on their intruded messages. To reduce the size of our model, we modify the open RCHAM and equip it with an alternative equivalence called *asynchronous bisimilarity*. The new chemical machine restricts intrusions to sets of messages that immediately trigger a reaction rule. Following the approach taken by Amadio et al. for the π-calculus [4], asynchronous bisimulation explicitly models asynchrony by modifying the requirement for the intrusions of messages. We prove that weak and asynchronous bisimilarities coincide, so that either notion may be selected in proofs.

Our last characterization of weak bisimilarity is given in terms of barbed bisimulation. The barbed congruence of Fournet and Gonthier [15] is strictly coarser than weak bisimilarity because the contexts of barbed congruence have no name matching capability, while the labels of weak bisimilarity separate names which exhibit the same behavior. This is the only difference between the two equivalences: by augmenting the join-calculus with a name matching operator, we show that (a variant of) barbed congruence then coincides with weak bisimilarity. This result relies on the technique presented in [16]; similar results for a variant of the π-calculus are given in [22].

Besides their use as proof techniques, our labeled semantics yield a better understanding of multiway synchronization in the join-calculus. They also provide a basis for comparing the join-calculus to other process calculi equipped with labeled transition systems, and in particular to the asynchronous π-calculus [11]. In these calculi, asynchrony means that message outputs have no continuation, and thus that there is no way to detect that a given message has been received. The usual weak bisimulation of the π-calculus has too much discriminating power, and separates processes with the same behavior such as $\mathsf{0}$ and $x(u).\bar{x}\langle u\rangle$; several remedies are considered in [21, 4]. Our bisimulations for the join-calculus embed similar accommodations, but they yield

simpler semantics because locality constrains interaction with the environment. This reduces the number of cases to consider and rules out processes in which the same message can be either received by the context or consumed by an internal reduction.

The paper is organized as follows. In Section 2, we introduce our language and its model. In Section 3, we define weak bisimulation and study its basic properties. In Section 4, we introduce asynchronous bisimulation and prove that it coincides with weak bisimulation. In Section 5, we give examples of bisimilar processes. In Section 6, we study the impact of name matching and relate our equivalences to barbed congruence. In Section 7, we discuss related work for the asynchronous π-calculus. We conclude in Section 8. Additional proofs appear in an appendix.

2. The open join-calculus

We define the open join-calculus and its operational semantics as extensions of the join-calculus and the RCHAM of Fournet and Gonthier [15]. We refer to previous works for a discussion of the join-calculus as a model of distributed programming [15, 17, 14].

2.1. Syntax and scopes

The syntax of the open join-calculus relies on a countable set of names $\mathcal{N}$ ranged over by x, y, u, v,...; tuples of names are written $u_i^{\,i\in 1..p}$ or simply $\tilde{u}$. The syntax includes processes P, open processes A, definitions D, and join-patterns J. A process P can be the inert process $\mathsf{0}$, a message $x\langle\tilde{u}\rangle$ sent on a name x that carries a tuple of names $\tilde{u}$, a parallel composition of processes, or a defining process $\mathsf{def}\ D\ \mathsf{in}\ P$ where D defines names that are local to P.

A definition D is a conjunction of reaction rules $J \triangleright P$ that associate join-patterns J to guarded processes P; the intended meaning is that, whenever messages match the pattern J, these messages can be replaced with a copy of the guarded process P where the formal parameters are replaced with the content of the messages. The only binders of the calculus are join-patterns, but the scope of names appearing in a join-pattern depends on their position: the scope of received names is the guarded process; the scope of names carrying messages is the main process of the definition and, recursively, all guarded processes of the definition (see Table 1).

An open process A is like a process except that it has open definitions at top-level instead of local ones. The open definition $\mathsf{def}_S\ D\ \mathsf{in}\ P$ exhibits a subset S of names defined by D that are visible from the environment: the *extruded names*. We may omit the index set S when it is empty and identify open definitions $\mathsf{def}_\emptyset\ D\ \mathsf{in}\ P$ with local definitions $\mathsf{def}\ D\ \mathsf{in}\ P$. We refer to the fragment of the calculus without extruded names as the *plain join-calculus*. We also write $\bigwedge_{i=1}^{n} J_i \triangleright P_i$ for $J_1 \triangleright P_1 \wedge \cdots \wedge J_n \triangleright P_n$ and $\prod_{i=1}^{n} A_i$ for $A_1 \mid \ldots \mid A_n$.

The *interface* of an open process A consists of two disjoint sets of names: free names $\mathrm{fv}(A)$ used in A to send messages out, and extruded names $\mathrm{xv}(A)$ used by the

Table 1
Syntax for the open join-calculus

P	$::=$	$\mathsf{0}$	A	$::=$	$\mathsf{0}$
	$\mid$	$x\langle u_i^{\,i\in 1..p}\rangle$		$\mid$	$x\langle u_i^{\,i\in 1..p}\rangle$
	$\mid$	$P \mid P$		$\mid$	$A \mid A$
	$\mid$	$\mathsf{def}\ D\ \mathsf{in}\ A$		$\mid$	$\mathsf{def}_S\ D\ \mathsf{in}\ A$
D	$::=$	$J \triangleright P$	J	$::=$	$x\langle u_i^{\,i\in 1..p}\rangle$
	$\mid$	$D \wedge D$		$\mid$	$J \mid J$

Table 2
Scopes for the open join-calculus

In join patterns

$\mathrm{rv}(x\langle\tilde{v}\rangle) \stackrel{\mathrm{def}}{=} \{u \in \tilde{v}\}$ $\quad$ $\mathrm{dv}(x\langle\tilde{v}\rangle) \stackrel{\mathrm{def}}{=} \{x\}$

$\mathrm{rv}(J \mid J') \stackrel{\mathrm{def}}{=} \mathrm{rv}(J) \uplus \mathrm{rv}(J')$ $\quad$ $\mathrm{dv}(J \mid J') \stackrel{\mathrm{def}}{=} \mathrm{dv}(J) \uplus \mathrm{dv}(J')$

In definitions

$\mathrm{dv}(J \triangleright P) \stackrel{\mathrm{def}}{=} \mathrm{dv}(J)$ $\quad$ $\mathrm{fv}(J \triangleright P) \stackrel{\mathrm{def}}{=} \mathrm{dv}(J) \cup (\mathrm{fv}(P) \setminus \mathrm{rv}(J))$

$\mathrm{dv}(D \wedge D') \stackrel{\mathrm{def}}{=} \mathrm{dv}(D) \cup \mathrm{dv}(D')$ $\quad$ $\mathrm{fv}(D \wedge D') \stackrel{\mathrm{def}}{=} \mathrm{fv}(D) \cup \mathrm{fv}(D')$

In processes

$\mathrm{fv}(A \mid A') \stackrel{\mathrm{def}}{=} (\mathrm{fv}(A) \cup \mathrm{fv}(A')) \setminus (\mathrm{xv}(A) \uplus \mathrm{xv}(A'))$ $\quad$ $\mathrm{fv}(\mathsf{0}) \stackrel{\mathrm{def}}{=} \emptyset$

$\mathrm{xv}(A \mid A') \stackrel{\mathrm{def}}{=} \mathrm{xv}(A) \uplus \mathrm{xv}(A')$ $\quad$ $\mathrm{xv}(\mathsf{0}) \stackrel{\mathrm{def}}{=} \emptyset$

$\mathrm{fv}(\mathsf{def}_S\ D\ \mathsf{in}\ A) \stackrel{\mathrm{def}}{=} (\mathrm{fv}(D) \cup \mathrm{fv}(A)) \setminus (\mathrm{dv}(D) \uplus \mathrm{xv}(A))$ $\quad$ $\mathrm{fv}(x\langle\tilde{v}\rangle) \stackrel{\mathrm{def}}{=} \{x, \tilde{v}\}$

$\mathrm{xv}(\mathsf{def}_S\ D\ \mathsf{in}\ A) \stackrel{\mathrm{def}}{=} S \uplus \mathrm{xv}(A)$ $\quad$ $\mathrm{xv}(x\langle\tilde{v}\rangle) \stackrel{\mathrm{def}}{=} \emptyset$

In solutions

$\mathrm{fv}(\mathscr{D} \vdash_S \mathscr{A}) \stackrel{\mathrm{def}}{=} (\mathrm{fv}(\mathscr{D}) \cup \mathrm{fv}(\mathscr{A})) \setminus (\mathrm{dv}(\mathscr{D}) \uplus \mathrm{xv}(\mathscr{A}))$

$\mathrm{xv}(\mathscr{D} \vdash_S \mathscr{A}) \stackrel{\mathrm{def}}{=} S \uplus \mathrm{xv}(\mathscr{A})$

environment to send messages in. We refer to names in either of these sets as *visible names*. Our notion of "free names" is thus more restrictive than usually; in [28], for instance, every visible name is considered "free". Other names that appear in a process are names bound in a join-pattern and not extruded; we refer to these names as *local names*. Received names $\mathrm{rv}(J)$, defined names $\mathrm{dv}(J)$ and $\mathrm{dv}(D)$, free names $\mathrm{fv}(D)$ and $\mathrm{fv}(A)$, and extruded names $\mathrm{xv}(A)$ are defined in Table 2. As usual, $\uplus$ means disjoint union.

In the whole paper, we identify terms that are equal up to a renaming of local names (called α-conversion) and we assume that all terms meet the following well-formed conditions. These conditions extend those of Fournet and Gonthier [15], and will be preserved by all transitions.

(1) Names carry fixed-sized messages, i.e., there is a recursive sorting discipline on names that avoids arity mismatch, in the style of Milner [27]. We refer to other works for a detailed treatment of sorts and arities in the join-calculus [14, 18].
(2) Join-patterns are linear, i.e., a variable may appear at most once in every join pattern. Linearity of received variables rules out name matching. Linearity of

defined variables does not affect expressiveness (cf. [14]) but it makes technical developments simpler.

(3) Sets of names extruded by different open sub-processes are disjoint – informally, these names are independently defined.

(4) Open definitions $\mathsf{def}_S\ D\ \mathsf{in}\ P$ define their extruded names ($S \subseteq \mathrm{dv}(D)$).

2.2. Open chemistry

We first illustrate our operational semantics for the process $\mathsf{def}_\emptyset\ x\langle\rangle \triangleright y\langle\rangle\ \mathsf{in}\ z\langle x\rangle$ (omitting structural equivalence). The interface contains no extruded name and two free names y, z. The message $z\langle x\rangle$ can be consumed by the environment, thus exporting x:

$$\mathsf{def}_\emptyset\, x\langle\rangle \triangleright y\langle\rangle\ \mathsf{in}\ z\langle x\rangle \xrightarrow{\{x\}z\langle x\rangle} \mathsf{def}_{\{x\}} x\langle\rangle \triangleright y\langle\rangle\ \mathsf{in}\ \mathsf{0}.$$

Once x is known by the environment, it cannot be considered local anymore – the environment can emit on x –, but it is not free either – the environment cannot modify or extend its definition. A new transition is enabled:

$$\mathsf{def}_{\{x\}} x\langle\rangle \triangleright y\langle\rangle\ \mathsf{in}\ 0 \xrightarrow{x\langle\rangle} \mathsf{def}_{\{x\}} x\langle\rangle \triangleright y\langle\rangle\ \mathsf{in}\ x\langle\rangle.$$

Now the process can input more messages on x, and also perform the two transitions below to consume the message on x and emit a message on y:

$$\mathsf{def}_{\{x\}} x\langle\rangle \triangleright y\langle\rangle\ \mathsf{in}\ x\langle\rangle \to \mathsf{def}_{\{x\}} x\langle\rangle \triangleright y\langle\rangle\ \mathsf{in}\ y\langle\rangle \xrightarrow{\{\}y\langle\rangle} \mathsf{def}_{\{x\}} x\langle\rangle \triangleright y\langle\rangle\ \mathsf{in}\ \mathsf{0}.$$

We now extend the RCHAM of Fournet and Gonthier [15] with extrusions, intrusions, and explicit bookkeeping of extruded names.

Definition 1. Open chemical solutions, ranged over by $\mathscr{S}, \mathscr{T}, \ldots$, are triples $(\mathscr{D}, S, \mathscr{A})$, written $\mathscr{D} \vdash_S \mathscr{A}$, where

- $\mathscr{D}$ is a multiset of definitions;
- S is a subset of names defined in $\mathscr{D}$ ($S \subseteq \mathrm{dv}(\mathscr{D})$);
- $\mathscr{A}$ is a multiset of open processes with disjoint sets of extruded names such that $\mathrm{dv}(\mathscr{D}) \cap \mathrm{xv}(\mathscr{A}) = \emptyset$.

The interface of an open solution $\mathscr{S}$ also consists of two disjoint sets of free names $\mathrm{fv}(\mathscr{S})$ and extruded names $\mathrm{xv}(\mathscr{S})$, defined in Table 2. (Functions $\mathrm{dv}(\cdot)$, $\mathrm{fv}(\cdot)$, and $\mathrm{xv}(\cdot)$ are extended to multisets of terms by taking unions for all terms in the multisets.) When no ambiguity arises, we identify the term A and the open solution $\emptyset \vdash_\emptyset A$. We also let $\mathscr{S} \mid P$ abbreviate the open solution $\mathscr{D} \vdash_S \mathscr{A}, P$, for all processes P and open solutions $\mathscr{D} \vdash_S \mathscr{A}$ obtained from $\mathscr{S}$ by α-conversion such that $\mathrm{fv}(P) \cap \mathrm{dv}(\mathscr{D}) \subseteq S$.

The chemical rules for the open RCHAM are given in Table 3; they define families of transitions between open solutions $\equiv$, $\to$, and $\xrightarrow{\alpha}$ where α ranges over labels of

Table 3
The open RCHAM

STR-NULL	$\vdash_S \mathbf{0} \equiv \vdash_S$
STR-PAR	$\vdash_S A \mid A' \equiv \vdash_S A, A'$
STR-AND	$D \wedge D' \vdash_S \equiv D, D' \vdash_S$
STR-DEF	$\vdash_S \mathsf{def}_{S'}\ D\ \mathsf{in}\ A \equiv D\sigma \vdash_{S \uplus S'} A\sigma$
RED	$J \triangleright P \vdash_S J\rho \to J \triangleright P \vdash_S P\rho$
EXT	$\vdash_S x\langle {v_i}^{i \in 1..p}\rangle \xrightarrow{S'\tilde{x}\langle {v_i}^{i \in 1..p}\rangle} \vdash_{S \cup S'}$
INT	$\vdash_{S \cup \{x\}} \xrightarrow{x\langle {v_i}^{i \in 1..p}\rangle} \vdash_{S \cup \{x\}} x\langle {v_i}^{i \in 1..p}\rangle$

Side conditions on the reacting solution $\mathscr{S} = (\mathscr{D} \vdash_S \mathcal{A})$:

STR-DEF	σ replaces $\mathrm{dv}(D) \backslash S'$ with distinct fresh names;
RED	ρ substitutes names for $\mathrm{rv}(J)$;
EXT	x is free, and $S' = \{v_i \mid i \in 1..p\} \cap (\mathrm{dv}(\mathscr{D}) \backslash S)$;
INT	${v_i}^{i \in 1..p}$ are either free, or fresh, or extruded.

the form $S\tilde{x}\langle\tilde{v}\rangle$ and $x\langle\tilde{v}\rangle$. By convention, each chemical rule mentions only the processes and definitions that are involved in the transition, and the transition applies to every chemical solution $\mathscr{S}$ that contains them. We define *structural equivalence* as the reflexive–symmetric–transitive closure of the structural moves given in Table 3. We write $\equiv \to \equiv$ for silent steps up to structural equivalence, and $\Rightarrow$ for series of such steps ($\Rightarrow \stackrel{\text{def}}{=} \bigcup_{n \geqslant 0} \equiv (\to \equiv)^n$).

Let us comment on the rules. The rules STR-NULL, STR-PAR, and STR-AND make parallel composition of processes and conjunction of definitions associative and commutative, with unit $\mathbf{0}$ for parallel composition. The rule STR-DEF enforces a lexical scoping discipline with scope-extrusion. The reduction rule RED states how messages can be consumed and replaced with a copy of a guarded process. These first five rules are those of the RCHAM, except that rule STR-DEF performs additional bookkeeping on extruded names. More precisely, the original RCHAM of Fournet and Gonthier [15] can be recovered as the restriction of the open RCHAM that operates on chemical solutions with no extruded variables and that does not use the rules EXT and INT.

The last two rules model interaction with the context. According to rule EXT, messages emitted on free names can be received by the environment; these messages export any defined name that was not previously known to the environment, thus causing the scope of its definition to be opened. This is made explicit by the set S' in the label of the transition $\xrightarrow{S'\tilde{x}\langle\tilde{v}\rangle}$. Names in S' must be distinct from any name that appears in the interface before the transition; once these names have been extruded they cannot be α-converted anymore, and behave like constants. Our rule resembles the OPEN rule for restriction in the π-calculus [28], with an important constraint due to locality: messages are either emitted on free names, to be consumed by EXT, or on names defined in the open solution, to be consumed by RED.

The rule INT enables the intrusion of messages on extruded names only. It can be viewed as a disciplined version of one of the two INPUT rules proposed by Honda and Tokoro for the asynchronous π-calculus, which enables the intrusion of any message [21]. The side condition of INT requires that intruded messages do not clash with local names of processes. (More implicitly, we may instead rely on the silent α-conversion on those local names; this is the original meaning of "intrusion" in [28].)

When applied to open solutions, our structural rules capture the intended meaning of extruded names: messages sent on extruded names can be moved inside or outside their defining process. For instance, we have the structural rearrangement

$$\vdash_S \; x\langle\tilde{v}\rangle \mid \mathsf{def}_{S'}\ D\ \mathsf{in}\ A \equiv \vdash_S \mathsf{def}_{S'}\ D\ \mathsf{in}\ (x\langle\tilde{v}\rangle \mid A)$$

for any extruded name x, and as long as the names in $\tilde{v}$ are not captured by D ($\{\tilde{v}\} \cap \mathrm{dv}(D) \subseteq S'$).

Remark 2. In its original presentation [15, 14], the join-calculus is equipped with auxiliary labeled transition systems that give an alternative, syntactic description of chemical reduction steps. Transitions are labeled with whole reaction rules; these large labels are discarded when the transition applies within a defining process that contains the same reaction rule. In contrast, our open RCHAM provides an extensional semantics of the calculus; labels are much simpler than definitions, and silent steps are not defined as a combination of labeled transitions and hiding.

2.3. Basic properties

In order to deal with bisimulations in the next section, we set additional notations and we state elementary properties of the transition system.

The following property is inherited from the RCHAM of Fournet and Gonthier [15]. It provides two different kinds of normal forms for open solutions.

Proposition 3. *Every open chemical solution is structurally equivalent to a solution that contains only simple reaction rules and messages, called a fully heated solution, of the form*

$$\{\ldots, J_j \triangleright P_j, \ldots\} \vdash_S \{\ldots, x_i\langle\tilde{u}_i\rangle, \ldots\}$$

which is unique up to α-conversion, and to a solution that contains a single open process of the form

$$\emptyset \vdash_\emptyset \left\{ \mathsf{def}_S \bigwedge J_j \triangleright P_j\ \mathsf{in}\ \prod x_i\langle\tilde{u}_i\rangle \right\}.$$

The multiset of messages $x_i\langle\tilde{u}_i\rangle$ can be further partitioned into three parts: messages on free names, messages on extruded names, and messages on locally defined names.

Given a fully heated solution $\mathscr{S} = \mathscr{D} \vdash_{S \uplus S'} \mathscr{M}$, the *restriction* of $\mathscr{S}$ on S is written $\mathscr{S}\backslash S$ and defined as follows:

$$(\mathscr{D} \vdash_{S \uplus S'} \mathscr{M})\backslash S \stackrel{\text{def}}{=} \mathscr{D} \vdash_{s'} \mathscr{M}.$$

More generally, the restriction operator $\backslash S$ is defined for all solutions and open processes that extrude every name of S, by applying the restriction to the structurally equivalent fully heated solution. Since $\backslash S$ is partially defined, when we write $\mathscr{S}\backslash S$ we will assume that $S \subseteq \mathrm{xv}(\mathscr{S})$. This linear, partial definition of restriction is consistent with our explicit management of input interface. In particular, it rules out erroneous restrictions on free names.

Definition 4. Let $\mathscr{S}$ be an open solution; a *global renaming* on $\mathscr{S}$ is a partial function from names to names with finite domain that is injective on $\mathrm{xv}(\mathscr{S})$.

Global renamings operate on open solutions (and similarly on open processes) by substituting their visible names. We let σ, σ' range over global renamings. As usual, the substitution may require α-conversion on local names to avoid clashes. Following the definition, global renamings may map free names onto previously free names, fresh names, or extruded names, but always map extruded names to distinct extruded names.

The next propositions gather basic facts about interfaces, global renamings, and chemical steps. Their proofs are straightforward consequences of previous definitions.

Proposition 5. *Let $\mathscr{S}$ be an open solution and $S \subseteq \mathrm{xv}(\mathscr{S})$.*

(1) *Let σ be a global renaming for $\mathscr{S}$. Then* $\mathrm{fv}(\mathscr{S}\sigma) = \mathrm{fv}(\mathscr{S})\sigma \backslash \mathrm{xv}(\mathscr{S})\sigma$.

(2) *Let $\mathscr{S} \equiv \mathscr{S}'$. Then*
   (a) $\mathrm{fv}(\mathscr{S}') = \mathrm{fv}(\mathscr{S})$ *and* $\mathrm{xv}(\mathscr{S}') = \mathrm{xv}(\mathscr{S})$;
   (b) $\mathscr{S}\backslash S \equiv \mathscr{S}'\backslash S$.

(3) *Let $\mathscr{S} \to \mathscr{S}'$. Then*
   (a) $\mathrm{fv}(\mathscr{S}') \subseteq \mathrm{fv}(\mathscr{S})$ *and* $\mathrm{xv}(\mathscr{S}') = \mathrm{xv}(\mathscr{S})$;
   (b) $\mathscr{S}\backslash S \equiv \to \equiv \mathscr{S}'\backslash S$.

The following proposition relates extrusions and intrusions to internal moves. It states two key properties of asynchrony: both extrusions and intrusions are "buffered", hence delayed extrusions and anticipated intrusions cannot affect internal steps:

Proposition 6. *Let $\mathscr{S}$ be an open solution.*

(1) $\mathscr{S} \equiv \xrightarrow{S\tilde{x}\langle\tilde{v}\rangle} \equiv \mathscr{S}'$ *if and only if* $\mathscr{S} \equiv (\mathscr{S}' \mid x\langle\tilde{v}\rangle)\backslash S$.
   If $\mathscr{S} \xrightarrow{S\tilde{x}\langle\tilde{v}\rangle} \Rightarrow \mathscr{S}'$, *then* $\mathscr{S} \Rightarrow \xrightarrow{S\tilde{x}\langle\tilde{v}\rangle} \equiv \mathscr{S}'$.

(2) $\mathscr{S} \xrightarrow{x\langle\tilde{v}\rangle} \mathscr{S}'$ *if and only if* $x \in \mathrm{xv}(\mathscr{S})$ *and* $\mathscr{S}' = \mathscr{S} \mid x\langle\tilde{v}\rangle$.
   If $\mathscr{S} \Rightarrow \xrightarrow{x\langle\tilde{v}\rangle} \mathscr{S}'$, *then* $\mathscr{S} \xrightarrow{x\langle\tilde{v}\rangle} \Rightarrow \mathscr{S}'$.

The next lemma details the correspondence between transitions in a chemical solution $\mathscr{S}$ and transitions in $\mathscr{S}\sigma$:

Lemma 7. *Let $\mathscr{S}$ be an open solution and σ be a global renaming for $\mathscr{S}$.*

(1) *$\mathscr{S} \equiv \mathscr{S}'$ if and only if $\mathscr{S}\sigma \equiv \mathscr{S}'\sigma$.*

(2) *If $\mathscr{S} \to \mathscr{T}$ then $\mathscr{S}\sigma \to \mathscr{T}\sigma$.*
If $\mathscr{S}\sigma \to \mathscr{T}$ then $\mathscr{S} \equiv \xrightarrow{S_1\overline{x_1}\langle\tilde{v}_1\rangle} \xrightarrow{y_1\langle\tilde{v}_1\rangle} \ldots \xrightarrow{S_n\overline{x_n}\langle\tilde{v}_n\rangle} \xrightarrow{y_n\langle\tilde{v}_n\rangle} \equiv \to \mathscr{S}'$ where $(\mathscr{S}' \setminus \biguplus_{i=1}^{n} S_i)\sigma \equiv \mathscr{T}$ and $x_i\sigma = y_i\sigma$ for $i = 1 \ldots n$.

(3) *If $\mathscr{S} \xrightarrow{x\langle\tilde{v}\rangle} \mathscr{T}$ then $\mathscr{S}\sigma \xrightarrow{(x\langle\tilde{v}\rangle)\sigma} \mathscr{T}\sigma$.*
If $\mathscr{S}\sigma \xrightarrow{y\langle\tilde{w}\rangle} \mathscr{T}$ then $\mathscr{S} \xrightarrow{x\langle\tilde{v}\rangle} \mathscr{S}'$ where $\mathscr{S}'\sigma = \mathscr{T}$ and $(x\langle\tilde{v}\rangle)\sigma = y\langle\tilde{w}\rangle$.

(4) *Let S be a set of names such that σ does not operate or range over S.*
If $\mathscr{S} \xrightarrow{S\tilde{x}\langle\tilde{v}\rangle} \mathscr{T}$ and $x\sigma \in \mathrm{fv}(\mathscr{S}\sigma)$, then $\mathscr{S}\sigma \xrightarrow{S\overline{x\sigma}\langle\tilde{v}\sigma\rangle} \mathscr{T}\sigma$.
If $\mathscr{S}\sigma \xrightarrow{S\tilde{y}\langle\tilde{w}\rangle} \mathscr{T}$, then $\mathscr{S} \xrightarrow{S\tilde{x}\langle\tilde{v}\rangle} \mathscr{S}'$ where $\mathscr{S}'\sigma = \mathscr{T}$ and $(x\langle\tilde{v}\rangle)\sigma = y\langle\tilde{w}\rangle$.

Proof. We show in detail only the second part of Case 2, which is the less obvious. The other cases are similar but easier. If $\mathscr{S}\sigma \to \mathscr{T}$, then by definition of rule RED (and after α-conversion) $\mathscr{S}$ and $\mathscr{T}$ must be of the form

$$\mathscr{S} = \mathscr{D}, J \triangleright P \vdash_S \mathscr{P}, M,$$

$$\mathscr{T} = \mathscr{D}\sigma, J\sigma \triangleright P\sigma \vdash_{S\sigma} \mathscr{P}_\sigma, P\sigma\rho,$$

where M is a parallel composition of messages and ρ is the substitution used in RED to consume the messages $M\sigma = J\sigma\rho$ in $\mathscr{S}\sigma$.

We partition messages in M according to the names carrying the messages. By definition of rule RED every such name in $M\sigma$ is defined in $J\sigma$, but before renaming these names may be either free or defined in J: in the case they are free, they are mapped to names defined in $J\sigma$ and extruded in $\mathscr{S}\sigma$. Let M'' be the parallel composition of messages in M sent on names defined in J, let n be the number of messages in M sent on free names, and assume that these messages are of the form $x_i\langle\tilde{v}_i\rangle$. Let also y_i be the unique names in $\mathrm{dv}(J)$ such that $x_i\sigma = y_i\sigma$. For $i = 0 \ldots n$ we write

$$\mathscr{S}_i \stackrel{\mathrm{def}}{=} \mathscr{D}, J \triangleright P \vdash_{S \cup \biguplus_{j=1}^{i-1} S_j} \mathscr{P}, M'', y_1\langle\tilde{v}_1\rangle, \ldots, y_{i-1}\langle\tilde{v}_{i-1}\rangle, x_i\langle\tilde{v}_i\rangle, \ldots, x_n\langle\tilde{v}_n\rangle,$$

where the sets S_i are successively defined as the names in $\tilde{v}_i$ that are not in the interface of $\mathscr{S}_{i-1}$. Using STR-PAR, we have $\mathscr{S} \equiv \mathscr{S}_0$. Applying EXT and INT, we have $\mathscr{S}_i \xrightarrow{S_i\tilde{x}_i\langle\tilde{v}_i\rangle} \xrightarrow{y_i\langle\tilde{v}_i\rangle} \mathscr{S}_{i+1}$. Using STR-PAR again, we assemble in $\mathscr{S}_n$ a parallel composition of messages $M' = J\rho'$ where ρ' operates on $\mathrm{rv}(J)$, $M'\sigma = M\sigma$, and $\rho'\sigma = \sigma\rho$. Applying RED, the process $P\rho'$ can then be substituted for M', yielding the solution $\mathscr{S}'$ of the lemma. □

Intrusions and extrusions always occur on disjoint sets of names, but this property is not preserved by global renaming. The next lemma describes how intrusions and extrusions may cancel one another after renaming.

Lemma 8. *Suppose* $\mathcal{S} \equiv \xrightarrow{S\tilde{x}\langle\tilde{v}\rangle} \Rightarrow \xrightarrow{y\langle\tilde{v}\rangle} \mathcal{S}'$, *and let* σ *be a global renaming for* $\mathcal{S}$ *such that* $y \notin S$ *and* $x\sigma = y\sigma$. *Then we have* $\mathcal{S}\sigma \Rightarrow (\mathcal{S}'\backslash S)\sigma$.

Proof. By applying Proposition 6, for some solution $\mathcal{T}$ we have

$$\mathcal{S} \equiv (\mathcal{T} \mid x\langle\tilde{v}\rangle)\backslash S \equiv \xrightarrow{S\tilde{x}\langle\tilde{v}\rangle} \xrightarrow{y\langle\tilde{v}\rangle} \mathcal{T} \mid y\langle\tilde{v}\rangle \Rightarrow \mathcal{S}'.$$

We carry over these transitions to $\mathcal{S}\sigma$ as follows:
- By applying Lemma 7(1), we have $\mathcal{S}\sigma \equiv ((\mathcal{T} \mid x\langle\tilde{v}\rangle)\backslash S)\sigma$.
- By hypothesis, we have $((\mathcal{T} \mid x\langle\tilde{v}\rangle)\backslash S)\sigma = ((\mathcal{T} \mid y\langle\tilde{v}\rangle)\backslash S)\sigma$.
- Since $\mathcal{T} \mid y\langle\tilde{v}\rangle \Rightarrow \mathcal{S}'$, we obtain $(\mathcal{T} \mid y\langle\tilde{v}\rangle)\backslash S \Rightarrow \mathcal{S}'\backslash S$ by Proposition 5(2,3) and $((\mathcal{T} \mid y\langle\tilde{v}\rangle)\backslash S)\sigma \Rightarrow (\mathcal{S}'\backslash S)\sigma$ by Lemma 7(1,2). □

3. Weak bisimulation

The join-calculus has been opened to support the standard notion of bisimulation [26]. This section defines weak bisimulation for open solutions and investigates its properties.

Definition 9. A relation ϕ on open solutions is a *weak simulation* if, whenever $\mathcal{S}\ \phi\ \mathcal{T}$, we have
(1) if $\mathcal{S} \equiv \rightarrow \equiv \mathcal{S}'$ then $\mathcal{T} \Rightarrow \mathcal{T}'$ and $\mathcal{S}'\ \phi\ \mathcal{T}'$;
(2) if $\mathcal{S} \equiv \xrightarrow{\alpha} \equiv \mathcal{S}'$ then $\mathcal{T} \Rightarrow \xrightarrow{\alpha} \Rightarrow \mathcal{T}'$ and $\mathcal{S}'\ \phi\ \mathcal{T}'$,
for all labels α of shape $x\langle\tilde{v}\rangle$ or $S\tilde{x}\langle\tilde{v}\rangle$ such that $\mathrm{fv}(\mathcal{T}) \cap S = \emptyset$.

A relation ϕ is a *weak bisimulation* when both ϕ and ϕ^{-1} are weak simulations. *Weak bisimilarity* $\approx$ is the largest weak bisimulation.

Following our conventions on processes as solutions, we shall write $A \approx B$ instead of $(\emptyset \vdash_\emptyset A) \approx (\emptyset \vdash_\emptyset B)$.

The simulation clause for extrusion does not consider labels whose set of extruded names S clashes with the free names of $\mathcal{T}$: such transitions can never be simulated. This standard technicality does not affect the intuitive discriminating power of bisimulation, because names in S can be α-converted before the extrusion.

It is possible to tell whether two processes are weakly bisimilar by reasoning on their synchronization trees [26], without the need to exhibit discriminating contexts. For example, $x\langle u\rangle \not\approx x\langle v\rangle$ because each process performs an extrusion with distinct labels. Likewise, $x\langle y\rangle \not\approx \mathsf{def}\ z\langle u\rangle \triangleright y\langle u\rangle\ \mathsf{in}\ x\langle z\rangle$ because the first process emits a free name (label $\bar{x}\langle y\rangle$) while the latter emits a local name that gets extruded (label $\{z\}\bar{x}\langle z\rangle$). Examples of processes that are weakly bisimilar are presented in Section 5.

Remark 10. Rule INT makes weak bisimulation sensitive to input interfaces: $\mathcal{S} \approx \mathcal{T}$ implies $\mathrm{xv}(\mathcal{S}) = \mathrm{xv}(\mathcal{T})$.

Proof. Assume $x \in \mathrm{xv}(\mathcal{S})$ and $\mathcal{S} \equiv \mathcal{S}'$ where $\mathcal{S}'$ is the fully heated solution obtained by Proposition 3. We have $x \in \mathrm{xv}(\mathcal{S}')$ by Proposition 5(2), so rule INT is enabled in $\mathcal{S}'$ and $\mathcal{S} \equiv \xrightarrow{x\langle \tilde{v} \rangle}$ for all fresh names $\tilde{v}$. The simulation of these transitions yields $\mathcal{T} \Rightarrow \mathcal{T}' \xrightarrow{x\langle \tilde{v} \rangle} \Rightarrow$, hence $x \in \mathrm{xv}(\mathcal{T}')$, and thus $x \in \mathrm{xv}(\mathcal{T})$ by Proposition 5(2, 3). □

3.1. Up to proof techniques

A whole range of "up to proof techniques" are available to reduce the size of the relation to exhibit when proving bisimilarities [26, 29, 34]. The lemma below establishes that our definition of weak bisimulation is robust with respect to reasoning up to structural equivalence, restriction of the input interface, and weak bisimilarity on the right. Its proof appears in the appendix. As usual, we derive the definitions of "weak bisimulation up to" from the definition of weak bisimulation, and we use the resulting definitions as proof techniques.

Lemma 11. *Let ϕ be a relation that satisfies all the bisimulation clauses of Definition 9 after replacing the requirement "$\mathcal{S}' \ \phi \ \mathcal{T}'$" with one of the following weaker requirements:*
(1) (*Up to structural equivalence*) $\mathcal{S}' \equiv \phi \equiv \mathcal{T}'$.
(2) (*Up to weak bisimilarity on the right*) $\mathcal{S}' \ \phi \approx \mathcal{T}'$.
(3) (*Up to restriction*) *there are a set of names S and two solutions $\mathcal{S}''$ and $\mathcal{T}''$ such that $\mathcal{S}' \equiv \mathcal{S}''\backslash S$, $\mathcal{T}' \equiv \mathcal{T}''\backslash S$, and $\mathcal{S}'' \ \phi \ \mathcal{T}''$.*
Then we have $\phi \subseteq \approx$.

3.2. Renaming and congruence properties

A context of the open join-calculus is an open process with a single process placeholder $[\cdot]$: we write $C[\cdot]$ for a context and $C[A]$ for the process obtained by substituting A for the placeholder. Whenever we apply a context, we implicitly assume that the resulting open process is well-formed. We sometimes use the subset of contexts that may bind some names of the interface of a process, but cannot prevent its execution: *evaluation contexts* $C[\cdot]$ are defined by the following grammar, up to structural equivalence:

$$C[\cdot] \stackrel{\text{def}}{=} [\cdot] \mid C[\cdot] \mid A \mid \mathsf{def}_S \ D \ \mathsf{in} \ C[\cdot],$$

A congruence is an equivalence that is preserved by application of all contexts.

Theorem 12. *Weak bisimilarity is a congruence.*

The proof is detailed in the appendix; its structure is almost generic to mobile process calculi in the absence of external choice (see, e.g., [21, 4] for the asynchronous π-calculus); it relies on two simpler closure properties:

Lemma 13. *Weak bisimilarity is preserved by application of evaluation contexts.*

Lemma 14. *Weak bisimilarity is preserved by global renaming.*

However, weak bisimulation up to global renaming is not a valid proof technique, because every intrusion could be cancelled by a substitution mapping fresh variables to extruded ones. For instance, let $A_n = \mathsf{def}_{\{y\}}\ y\langle\rangle \triangleright a\langle\rangle\ \mathsf{in}\ \prod_{i=1}^{n} x_i\langle\rangle$, let $B_n = \mathsf{def}_{\{y\}}\ y\langle\rangle \triangleright b\langle\rangle\ \mathsf{in}\ \prod_{i=1}^{n} x_i\langle\rangle$, and let $\phi = \{(A_n, B_n) \mid n \geqslant 0\}$. The processes A_n and B_n are not bisimilar because, after an intrusion on $y\langle\rangle$ and a silent step, A_n sends $a\langle\rangle$ while B_n sends $b\langle\rangle$. However, the relation ϕ is a weak bisimulation up to the renaming $\{{}^{y}/_{x_{n+1}}\}$ after this intrusion.

4. Asynchronous bisimulation

In order to prove that two processes are bisimilar, a large candidate bisimulation can be a nuisance, as it requires the analysis of numerous transition cases. Unfortunately, weak bisimulations on open chemical solutions are typically rather large. For example, a process with an extruded name has infinitely many derivatives even if no "real" computation is ever performed. Consider the equivalence

$$\mathsf{def}\ x\langle u\rangle \mid y\langle v\rangle \triangleright P\ \mathsf{in}\ z\langle x\rangle \approx \mathsf{def}\ x\langle u\rangle \mid y\langle v\rangle \triangleright Q\ \mathsf{in}\ z\langle x\rangle.$$

These two processes are bisimilar because their join-pattern cannot be triggered, regardless of the messages the environment may send on x. Still, one is confronted with infinite models on both sides, with a distinct chemical solution for every multiset of messages that have been intruded on x so far. This problem with weak bisimulation motivates an alternative formulation.

We refine the open RCHAM by allowing inputs only when they immediately trigger a guarded process. For example, the two processes above become inert after an extrusion $\{x\}\bar{z}\langle x\rangle$, hence trivially bisimilar. If we applied this refinement with the same labels for input as before, however, we would obtain a dubious result. The solution $x\langle\rangle|y\langle\rangle|z\langle\rangle \triangleright P \vdash_{\{x,y\}} z\langle\rangle$ can progress by first inputing two messages $x\langle\rangle$ and $y\langle\rangle$, then performing a silent step that consumes these two messages together with the local message $z\langle\rangle$ already in the solution. Yet, neither $x\langle\rangle$ nor $y\langle\rangle$ alone can trigger the process P, and therefore this solution would become inert, too. This suggests the use of *join*-inputs on x and y in transitions such as

$$x\langle\rangle \mid y\langle\rangle \mid z\langle\rangle \triangleright P \vdash_{\{x,y\}} z\langle\rangle \xrightarrow{x\langle\rangle \mid y\langle\rangle} x\langle\rangle \mid y\langle\rangle \mid z\langle\rangle \triangleright P \vdash_{\{x,y\}} P.$$

Table 4
The J-open RCHAM

Rules STR-(NULL,PAR,AND,DEF) and EXT are as in Table 3.	
INT-J	$J \triangleright P \vdash_S M \xrightarrow{M'} J \triangleright P \vdash_S P\rho$
Side condition:	$J\rho \equiv M \mid M'$, $\mathrm{dom}(\rho) = \mathrm{rv}(J)$, $\mathrm{dv}(M') \subseteq S$, names in $\mathrm{rv}(M')$ are either free, or fresh, or extruded.

On the other hand, the solution $x\langle\rangle \mid y\langle\rangle \mid z\langle\rangle \triangleright P \vdash_{\{x\}} z\langle\rangle$ is truly inert, since the environment has no access to y, and thus cannot trigger P. In this case, our refinement suppresses all input transitions.

4.1. The J-open RCHAM

The J-open RCHAM is defined in Table 4 as a replacement for the intrusion rule. In contrast with the rule INT of Table 3, the new rule INT-J permits the intrusion of messages only if these messages are immediately used to trigger a process. This is formalized by allowing labels M' that are parallel compositions of messages. If the solution contains a complementary process M such that the combination $M \mid M'$ matches the join-pattern of a reaction rule, then the transition occurs and triggers this reaction rule. As for INT, we restrict intrusions in M' to messages on extruded names.

We identify intrusions in the case $M' = \mathsf{0}$ with silent steps; the rule RED is thus omitted from the new chemical machine. Nonetheless, we maintain the distinction between internal moves and proper input moves in the discussion.

Each chemical solution now has two different models: for instance, the solution $x\langle\rangle \mid y\langle\rangle \triangleright P \vdash_{\{x\}}$ has no transition in the J-open RCHAM, while it has infinite series of transitions $\xrightarrow{x\langle\rangle} \xrightarrow{x\langle\rangle} \xrightarrow{x\langle\rangle} \cdots$ in the open RCHAM. In the sequel, we shall keep the symbol $\xrightarrow{\alpha}$ for the open RCHAM and use $\xrightarrow{\alpha}_{\mathrm{J}}$ for the J-open RCHAM; we may drop the subscript J when no ambiguity can arise.

As a direct consequence of their definitions, we have the following relation between the two models.

Proposition 15. *Let $\mathscr{S}$ be an open solution.*

(1) *If $\mathscr{S} \xrightarrow{x_1\langle\tilde{v}_1\rangle \mid \cdots \mid x_n\langle\tilde{v}_n\rangle}_{\mathrm{J}} \mathscr{T}$, then $\mathscr{S} \xrightarrow{x_1\langle\tilde{v}_1\rangle} \cdots \xrightarrow{x_n\langle\tilde{v}_n\rangle} \equiv \rightarrow \mathscr{T}$.*

(2) *If $\mathscr{S} \mid x\langle\tilde{u}\rangle \equiv \xrightarrow{M}_{\mathrm{J}} \equiv \mathscr{T}$ and $x \in \mathrm{xv}(\mathscr{S})$, then*

(a) *either $\mathscr{S} \equiv \xrightarrow{M}_{\mathrm{J}} \equiv \mathscr{S}'$ with $\mathscr{S}' \mid x\langle\tilde{u}\rangle \equiv \mathscr{T}$;*

(b) *or $\mathscr{S} \equiv \xrightarrow{M \mid x\langle\tilde{u}\rangle}_{\mathrm{J}} \equiv \mathscr{T}$.*

(3) *$\mathscr{S} \xrightarrow{S\tilde{x}\langle\tilde{v}\rangle}_{\mathrm{J}} \mathscr{T}$ if and only if $\mathscr{S} \xrightarrow{S\tilde{x}\langle\tilde{v}\rangle} \mathscr{T}$*

Proof. (1) The side-conditions on names that may appear on intrusion labels are the same for the two RCHAMS, so we can use rule INT to intrude every message $x_i\langle\tilde{v}_i\rangle$ of the

compound label $x_1\langle\tilde{v}_1\rangle \mid \cdots \mid x_n\langle\tilde{v}_n\rangle$ one at a time. Once this is done, we can use the structural rule STR-PAR to assemble the parallel composition of messages that matches the join-pattern used in INT-J, and perform a RED transition that uses the same reaction rule to consume these messages and triggers the same process as in INT-J.

(2) The choice between the two cases depends on whether the message $x\langle\tilde{u}\rangle$ is consumed by INT-J.
   (a) For any series of reductions $\mathscr{S} \mid x\langle\tilde{u}\rangle \equiv \xrightarrow{M} \mathscr{T}$ such that the intrusion does not consume the message $x\langle\tilde{u}\rangle$, any preliminary structural step that operates on this message either commutes with the intrusion or is reverted before the intrusion, so we can build another series $\mathscr{S} \mid x\langle\tilde{u}\rangle \equiv \xrightarrow{M} \mathscr{S}' \mid x\langle\tilde{u}\rangle \equiv \mathscr{T}$ in which every step from $\mathscr{S}$ to $\mathscr{S}'$ applies independently of the presence of $x\langle\tilde{u}\rangle$.
   (b) When $x\langle\tilde{u}\rangle$ is consumed, by definition of INT-J, the transitions of the lemma can be decomposed into

$$\mathscr{S} \equiv \mathscr{D} \vdash_S \mathscr{P}, N \mid x\langle\tilde{u}\rangle \xrightarrow{M} \mathscr{T}' \equiv \mathscr{T},$$

   where $N \mid x\langle\tilde{u}\rangle$ is the process consumed by INT-J. We also have the transition $\mathscr{D} \vdash_S \mathscr{P}, N \xrightarrow{M \mid x\langle\tilde{u}\rangle} \mathscr{T}'$ and, as described above, we can eliminate preliminary structural steps that affect the message $x\langle\tilde{u}\rangle$.

(3) Extrusions share the same definition. □

4.2. Asynchronous bisimulation

Next, we adapt the definition of weak bisimulation (Definition 9) to the new J-open RCHAM. Consider for instance the two processes:

$$P \stackrel{\text{def}}{=} \mathsf{def}\ x\langle\rangle \triangleright a\langle\rangle \wedge a\langle\rangle \mid y\langle\rangle \triangleright R \mathsf{in}\ z\langle x, y\rangle,$$
$$Q \stackrel{\text{def}}{=} \mathsf{def}\ x\langle\rangle \mid y\langle\rangle \triangleright R \mathsf{in}\ z\langle x, y\rangle$$

and assume $a \notin \mathrm{fv}(R)$. With the initial open RCHAM, the processes P and Q are weakly bisimilar. With the new J-open RCHAM and the same definition of weak bisimulation, this does not hold because P can input $x\langle\rangle$ after emitting on z while Q cannot. But if we consider the weak bisimulation that uses join-input labels instead of single ones, Q can input $x\langle\rangle|y\langle\rangle$ while P cannot, and P and Q are still separated. It turns out that weak bisimulation discriminates too much in the J-open RCHAM.

In order to retain an asynchronous semantics, weak bisimulation must be relaxed, so that a process may simulate an INT-J transition even if it does not immediately consume all its messages. This leads us to the following definition.

Definition 16. A relation ϕ on open solutions is an *asynchronous simulation* if, whenever $\mathscr{S}\ \phi\ \mathscr{T}$, we have

(1) if $\mathscr{S} \equiv \xrightarrow{S\bar{x}\langle\tilde{v}\rangle} \equiv \mathscr{S}'$ then $\mathscr{T} \Rightarrow \xrightarrow{S\bar{x}\langle\tilde{v}\rangle} \Rightarrow \mathscr{T}'$ and $\mathscr{S}'\ \phi\ \mathscr{T}'$
   (for all labels $S\bar{x}\langle\tilde{v}\rangle$ such that $\mathrm{fv}(\mathscr{T}) \cap S = \emptyset$;

(2) if $\mathscr{S} \equiv \xrightarrow{M} \equiv \mathscr{S}'$, then $\mathscr{T} \mid M \Rightarrow \mathscr{T}'$ and $\mathscr{S}' \ \phi \ \mathscr{T}'$;
(3) $\mathrm{xv}(\mathscr{S}) = \mathrm{xv}(\mathscr{T})$.

A relation ϕ is an *asynchronous bisimulation* when both ϕ and ϕ^{-1} are asynchronous simulations. *Asynchronous bisimilarity* $\approx_{\mathrm{a}}$ is the largest asynchronous bisimulation.

In the definition above, the usual clause for silent steps is omitted (it is subsumed by the clause for intrusions with $M = \mathsf{0}$). On the other hand, an additional clause requires that related solutions have the same extruded names. Otherwise, Remark 10 would not hold for asynchronous bisimulation and, for instance, the open deadlocked solution $x\langle\rangle \mid y\langle\rangle \triangleright P \vdash_{\{y\}}$ would be equivalent to the empty solution $\vdash_{\emptyset}$.

We now establish that asynchronous bisimilarity and weak bisimilarity actually coincide, by relating their respective intrusion clauses.

Theorem 17. $\approx_{\mathrm{a}} = \approx$.

In the following, we use up to proof techniques (up to structural equivalence, up to asynchronous bisimulation on the right, up to restriction); these techniques can be defined and validated in the same style as Lemma 11. We also rely on a simple closure property, proved in the appendix:

Proposition 18. *If* $\mathscr{S} \approx_{\mathrm{a}} \mathscr{T}$ *and* $x \in \mathrm{xv}(\mathscr{S})$, *then* $\mathscr{S} \mid x\langle\tilde{v}\rangle \ \approx_{\mathrm{a}} \ \mathscr{T} \mid x\langle\tilde{v}\rangle$.

Proof of Theorem 17. We prove the inclusions $\approx \subseteq \approx_{\mathrm{a}}$ and $\approx_{\mathrm{a}} \subseteq \approx$. We consider only the intrusions, as all other transitions are identical.

Weak bisimilarity is an asynchronous bisimulation. Suppose $\mathscr{S} \approx \mathscr{T}$. According to Proposition 15(1), for every join-intrusion $\mathscr{S} \xrightarrow{M} \mathscr{S}'$ with label $M = x_1\langle\tilde{v}_1\rangle \mid \cdots \mid x_n\langle\tilde{v}_n\rangle$ in the J-open RCHAM, there is a series of transitions in the open machine that leads to the same solution $\mathscr{S}'$:

$$\mathscr{S} \xrightarrow{x_1\langle\tilde{v}_1\rangle} \cdots \xrightarrow{x_n\langle\tilde{v}_n\rangle} \equiv \ \rightarrow \mathscr{S}'.$$

By weak bisimulation hypothesis, we obtain a mixed series of internal moves and single intrusions from $\mathscr{T}$:

$$\mathscr{T} \Rightarrow \xrightarrow{x_1\langle\tilde{v}_1\rangle} \Rightarrow \cdots \Rightarrow \xrightarrow{x_n\langle\tilde{v}_n\rangle} \Rightarrow \mathscr{T}'.$$

with $\mathscr{S}' \approx \mathscr{T}'$. In the open RCHAM we can always perform intrusions before internal reductions: by iterating Proposition 6(2) we obtain

$$\mathscr{T} \xrightarrow{x_1\langle\tilde{v}_1\rangle} \cdots \xrightarrow{x_n\langle\tilde{v}_n\rangle} \equiv \mathscr{T} \mid M \Rightarrow \mathscr{T}'$$

and we obtain the join-intrusion requirement of Definition 16 for the series of reduction $\mathscr{T} \mid M \Rightarrow \mathscr{T}'$.

Asynchronous bisimilarity is a weak bisimulation. Suppose $\mathscr{S} \equiv \xrightarrow{x\langle\tilde{v}\rangle} \equiv \mathscr{S}'$ and $\mathscr{S} \approx_{\mathrm{a}} \mathscr{T}$. By definition of rule INT, $x \in \mathrm{xv}(\mathscr{S})$ and $\mathscr{S}' \equiv \mathscr{S} \mid x\langle\tilde{v}\rangle$. By asynchronous

bisimulation hypothesis, $\mathrm{xv}(\mathscr{S}) = \mathrm{xv}(\mathscr{T})$, thus $x \in \mathrm{xv}(\mathscr{T})$ and $\mathscr{T} \equiv \xrightarrow{x\langle \tilde{v} \rangle} \equiv \mathscr{T} \mid x\langle \tilde{v} \rangle$. We conclude by Proposition 18. □

4.3. Bisimulation up to evaluation context

As an illustration, we show that $P \approx_{\mathrm{a}} Q$ where P and Q are the processes at the beginning of Section 4.2. Both P and Q perform the same extrusion labeled $\{x, y\}\bar{z}\langle x, y \rangle$, therefore it suffices to prove

$$A \stackrel{\mathrm{def}}{=} \mathsf{def}_{\{x,y\}} x\langle\rangle \triangleright a\langle\rangle \wedge a\langle\rangle \mid y\langle\rangle \triangleright R \ \mathsf{in}\ \mathsf{0} \approx_{\mathrm{a}} B \stackrel{\mathrm{def}}{=} \mathsf{def}_{\{x,y\}} x\langle\rangle \mid y\langle\rangle \triangleright R \ \mathsf{in}\ \mathsf{0}.$$

To this end, we state the bisimulation up to evaluation context proof technique for asynchronous bisimulations. This technique is formalized in [34], and provides an effective tool for establishing equivalences. (This technique does not directly apply to weak bisimulation, because intrusions could always be cancelled by discarding the intruded messages. For instance the two processes $\mathsf{def}_{\{x\}}\ x\langle\rangle \triangleright a\langle\rangle\ \mathsf{in}\ \mathsf{0}$ and $\mathsf{def}_{\{x\}}\ x\langle\rangle \triangleright b\langle\rangle\ \mathsf{in}\ \mathsf{0}$ are not bisimilar, but they are bisimilar up to parallel composition, because the context $x\langle\rangle \mid [\,\cdot\,]$ can be discarded after intrusion but before the guarded processes are triggered.)

Lemma 19 (Up to evaluation contexts). *Let ϕ be a relation on open processes that satisfies all the clauses of Definition* 16 *after replacing the requirement "$\mathscr{S}'\ \phi\ \mathscr{T}'$" with "there is an evaluation context $C[\,\cdot\,]$ such that $\mathscr{S}' \equiv C[A]$, $\mathscr{T}' \equiv C[B]$, and $A\ \phi\ B$". We have $\phi \subseteq \approx_{\mathrm{a}}$.*

Proof. Let ϕ' be the relation that contains all pairs of well-formed processes $(C[A], C[B])$ such that $A\ \phi\ B$ and $C[\,\cdot\,]$ is an evaluation context. We show that ϕ' is an asynchronous bisimulation up to restriction and structural equivalence, and remark that $\phi \subseteq \phi' \subseteq \approx_{\mathrm{a}}$. Let $C[A]\ \phi'\ C[B]$.

Assume $C[A] \xrightarrow{S\tilde{x}\langle \tilde{v} \rangle} \mathscr{T}$ with $S \cap \mathrm{fv}(C[B]) = \emptyset$. If the extruded message is in $C[\,\cdot\,]$, this extrusion directly corresponds to one of $C[B]$; this yields related processes of the form $C'[A]\ \phi'\ C'[B]$. If the extruded message is in A, there is an extrusion $A \xrightarrow{S'\tilde{x}\langle \tilde{v} \rangle} A'$, with $S' \subseteq S$ and $\mathscr{T} \equiv C'[A']$. (Names in $S \backslash S'$ are extruded names bound in $C[\,\cdot\,]$.) This extrusion is simulated by $B \Rightarrow \xrightarrow{S'\tilde{x}\langle \tilde{v} \rangle} \Rightarrow B'$ with $A'\ \phi\ B'$, hence $C[B] \Rightarrow \xrightarrow{S\tilde{x}\langle \tilde{v} \rangle} \Rightarrow C'[B']$ with $C'[A']\ \phi'\ C'[B']$.

Assume $C[A] \xrightarrow{M} \mathscr{T}$. The names $\mathrm{dv}(M)$ are extruded by the same definition. If the definition is in A (including the case of steps internal to A), we also have an intrusion $A \xrightarrow{M'} A'$ with $M' \equiv M \mid M''$ and $\mathscr{T} \equiv C'[A']$ (messages in M'' account for additional messages provided by $C[\,\cdot\,]$). We have $B \mid M \mid M'' \Rightarrow B'$, hence $C[B] \Rightarrow C'[B']\ \phi'\ C'[A'] \equiv \mathscr{T}$. If the definition is in $C[\,\cdot\,]$ (including other cases of steps internal to $C[A]$), the intrusion can be simulated from $C[B]$ after extracting from B every message of A that is consumed by this transition. Let M' be the messages of A consumed in this intrusion. For every message of M', A can perform an extrusion, and B can

simulate this extrusion. Let $A' \phi B'$ be the processes obtained after performing all these extrusions, and let S collect all the extruded names. We have $\mathscr{T} \equiv C[A']\backslash S$ and $C[B] \Rightarrow C[B' \mid M']\backslash S \equiv \stackrel{M}{\rightarrow} \equiv C'[B']\backslash S$. Up to restriction, $C'[A']$ and $C'[B']$ are thus related by ϕ'. □

Proof of the example. Let $A_n \stackrel{\text{def}}{=} A \mid \prod_{i=1}^{n} a\langle\rangle$ and $B_n \stackrel{\text{def}}{=} B \mid \prod_{i=1}^{n} x\langle\rangle$. We prove that the relation $\phi \stackrel{\text{def}}{=} \{(A_n, B_n) \mid n \geqslant 0\}$ is an asynchronous bisimulation up to evaluation context and structural equivalence. All the initial transitions of the J-open RCHAM for A_n and B_n are intrusions on x and y:

(1) $A_n \equiv \xrightarrow{x\langle\rangle} \equiv A_{n+1}$,

(2) $B_n \equiv \xrightarrow{x\langle\rangle \mid y\langle\rangle} \equiv R \mid B_n$,

(3) $A_{n+1} \equiv \xrightarrow{y\langle\rangle} \equiv R \mid A_n$,

(4) $B_{n+1} \equiv \xrightarrow{y\langle\rangle} \equiv R \mid B_n$.

Intrusions (3) and (4) lead to processes in ϕ up to the context $R \mid [\cdot]$. Intrusions (1) do not correspond to any intrusion in B_n, but nonetheless the asynchronous bisimulation requirement can be met by adding the message $x\langle\rangle$ in parallel to B_n; we obtain two related processes at rank $n+1$. Intrusions (2) do not correspond to single intrusions in A_n, but the bisimulation requirement can be met after adding the two messages in parallel to A_n, since two internal steps can first consume the message $x\langle\rangle$ and release a message $a\langle\rangle$ instead, then consume $a\langle\rangle \mid y\langle\rangle$ and trigger R. □

4.4. Ground bisimulations

Ground bisimulation is a variant of bisimulation obtained by restricting the intrusions to labels that convey fresh names. As first observed in the π-calculus [21, 4, 10], asynchrony brings another interesting property as regards the number of transitions to consider: the ground variant of bisimilarity coincides with the original one. This property also holds in the join-calculus, thus providing proof techniques with, for every chemical solution, exactly one intrusion per extruded name when using weak bisimulation, and one intrusion per "active" partial join-pattern when using asynchronous bisimulation. The proof appears in the appendix.

Proposition 20. *Let $\approx_{\mathrm{g}}$ be the bisimilarity obtained from Definitions* 9 *after restricting intrusion labels to labels of the form $x\langle\tilde{v}\rangle$ where the names $\tilde{v}$ are pairwise-distinct names that do not appear in the solution. Similarly, let $\approx_{\mathrm{ag}}$ be the bisimilarity obtained from Definition* 16 *by restricting intrusion labels to $x_1\langle\tilde{v}\rangle \mid \ldots \mid x_m\langle\tilde{v}\rangle$ where the names $\tilde{v}$ are pairwise-distinct fresh names. We have $\approx_{\mathrm{g}} = \approx = \approx_{\mathrm{a}} = \approx_{\mathrm{ag}}$.*

5. Examples

We present a collection of simple bisimilarities, together with their proofs and applications. These equations crucially rely on locality. We refer to Fournet [14] for

more complex applications of labeled bisimulation proofs, and to Nestmann [31] for an investigation of join-patterns without locality.

In the join-calculus, whenever a name x is defined by a single clause of the form $x\langle\tilde{y}\rangle \triangleright P$, the outcome of any message sent on x is determined, independently of the context.

Example 21 (Deterministic reductions).

$$\mathsf{def}_{\{x\}} x\langle\tilde{y}\rangle \triangleright P \ \mathsf{in}\ x\langle\tilde{u}\rangle \approx P\{\tilde{u}/\tilde{y}\} \mid \mathsf{def}_{\{x\}} x\langle\tilde{y}\rangle \triangleright P \ \mathsf{in}\ \mathsf{0}.$$

This simple equation is especially useful, as it handles most of the reduction steps that occur in standard deterministic encodings, such as the encodings of functions and of data structures.

Proof. Let $A = \mathsf{def}_{\{x\}}\ x\langle\tilde{y}\rangle \triangleright P\ \mathsf{in}\ \mathsf{0}$, and let ϕ be the singleton relation $(A \mid x\langle\tilde{u}\rangle, A \mid P\{^{\tilde{u}}/_{\tilde{y}}\})$. We prove that the reflexive closure of ϕ is an asynchronous bisimulation up to evaluation context and structural equivalence. The process $A \mid x\langle\tilde{u}\rangle$ has two transitions: (1) The internal step

$$A \mid x\langle\tilde{u}\rangle \equiv \rightarrow \equiv A \mid P\{\tilde{u}/\tilde{y}\}$$

yields the related process; all transitions of $A \mid P\{^{\tilde{u}}/_{\tilde{y}}\}$ can thus be simulated by $A \mid x\langle\tilde{u}\rangle$ by composing this internal step and the same transitions, yielding identical processes. (2) The intrusions

$$A \mid x\langle\tilde{u}\rangle \equiv \stackrel{x\langle\tilde{v}\rangle}{\rightarrow} \equiv A \mid x\langle\tilde{u}\rangle \mid P\{\tilde{u}/\tilde{y}\}$$

are simulated by identical intrusions on the other side; these resulting processes can be obtained from ϕ by applying the context $[\,\cdot\,] \mid P\{^{\tilde{v}}/_{\tilde{y}}\}$. □

The following examples show that weak bisimulation is largely insensitive to the shape of join-patterns. To begin with, some straightforward equations allows one to get rid of redundant rules in definitions:

Example 22 (Redundant definitions).

$$\mathsf{def}_S\ D \wedge D' \ \mathsf{in}\ A \approx \mathsf{def}_S\ D \wedge D \wedge D' \ \mathsf{in}\ A, \tag{1}$$

$$\mathsf{def}_S\ J \triangleright P \wedge J' \triangleright P' \wedge D' \ \mathsf{in}\ A \approx \mathsf{def}_S\ J \triangleright P \wedge J' \triangleright P' \wedge J \mid J' \triangleright P \mid P' \wedge D' \ \mathsf{in}\ A. \tag{2}$$

Similarly, definitions can be simplified whenever some of their locally defined names do not occur anywhere else. The following example shows that it is not possible to observe the internal state of processes.

Example 23 (Adjunction of internal state).

$$\mathsf{def}_S\ J \triangleright P\ \mathsf{in}\ A \approx \mathsf{def}_S\ J \mid s\langle\tilde{v}\rangle \triangleright P \mid s\langle\tilde{v}\rangle \mathsf{in}\ A \mid s\langle\tilde{v}\rangle,$$

where s is a fresh name and $\{\tilde{v}\} \cap \mathrm{rv}(J) = \emptyset$.

Indeed, there is always one available message $s\langle\tilde{v}\rangle$ that conveys the same names, which are initially bound in a process *within the same scope*. This equality suffices to prove interesting properties with regards to our scoping rules. If we take $\{\tilde{v}\} = \mathrm{dv}(J) \cup \{s\}$, then all occurrences of names of $\mathrm{dv}(J)$ that appear in P – we call them *recursive occurrences* – are now bound as received variables. Up to weak bisimulation, we can therefore eliminate recursion from every definition. If we take $\{\tilde{v}\} = (\mathrm{fv}(P) \cup \{s\}) \backslash \mathrm{rv}(J)$, then all names in P are now bound as received variables. This is reminiscent of lambda-lifting in the λ-calculus; in combination with the congruence property of weak bisimilarity, this validates a compilation scheme for the join-calculus that would replace every process with an equivalent process with simpler binders (either receptions or immediate definitions).

Next, we consider the introduction of intermediate buffers; as one would expect in an asynchronous calculus, our semantics is not sensitive to such buffers, either before a join-synchronization or after it.

Example 24 (Buffering before synchronization).

$$\mathsf{def}_S\ D\ \mathsf{in}\ A \approx \mathsf{def}_S\ x\langle\tilde{u}\rangle \triangleright x'\langle\tilde{u}\rangle \wedge D'\ \mathsf{in}\ A,$$

where $x \in \mathrm{dv}(D)$, x' is a fresh name, and D' is obtained from D by substituting $x'\langle\tilde{v}\rangle$ for $x\langle\tilde{v}\rangle$ in every join-pattern.

Example 25 (Buffering after synchronization).

$$\mathsf{def}_S\ J \triangleright P \wedge D\ \mathsf{in}\ A \approx \mathsf{def}_S\ J \triangleright x\langle\tilde{v}\rangle \wedge x\langle v\rangle P \wedge D\ \mathsf{in}\ A,$$

where x is a fresh name and $\tilde{v}$ is a tuple that conveys the names of $\mathrm{rv}(J)$.

These simple properties of asynchrony are easily established. Using a weak bisimulation argument, for instance, we establish the equation above.

Proof of Example 25. Let ϕ be the relation that contains all pairs of open processes (A_1, A_2) of the form

$$A_1 \stackrel{\mathrm{def}}{=} \mathsf{def}_S\ J \triangleright P \wedge D\ \mathsf{in}\ A \mid \prod\nolimits_{\rho \in U} P\rho,$$

$$A_2 \stackrel{\mathrm{def}}{=} \mathsf{def}_S\ J \triangleright x\langle\tilde{v}\rangle \wedge x\langle\tilde{v}\rangle \triangleright P \wedge D\ \mathsf{in}\ A \mid \prod\nolimits_{\rho \in U} x\langle\tilde{v}\rho\rangle$$

for the terms D and P of the lemma, and for all $S \subseteq \mathrm{dv}(D)$, A, and U such that x is fresh and U is a finite multiset of substitutions with domain $\tilde{v}$. We check that ϕ is a

weak bisimulation up to structural equivalence, and obtain the lemma in the case U is empty. We detail only the transitions that affect U.

- Any transition that consumes a message $x\langle\tilde{v}\rho\rangle$ in A_2 is simulated in A_1 with no transition, leading to a pair of related processes with $A \mid P\rho$ instead of A and a smaller multiset U.
- Any transition that involves processes $P\rho$ in A_1 can be simulated in A_2 after performing silent steps that substitute these processes $P\rho$ for the corresponding messages $x\langle\tilde{v}\rho\rangle$. After structural rearrangement, the derivatives of A_1 and A_2 are still in ϕ for some other choice of S, A, and U.
- Any transition that uses the join-patterns $J \triangleright P$ in A_1 or $J \triangleright x\langle\tilde{v}\rangle$ in A_2, respectively, is handled by adding the substitution used by rule RED to U. □

The buffering of partial join-patterns may not preserve weak bisimilarity, because it can affect the branching structure of processes. For instance, the internal commitment to one of the two messages on x separates the second process from the first one:

$$\begin{aligned} &\mathsf{def}_{\{z\}}\ x\langle u\rangle \mid y\langle\rangle \mid z\langle v\rangle \triangleright P && \mathsf{in}\ x\langle 1\rangle \mid x\langle 2\rangle \mid y\langle\rangle \\ \not\approx\ &\mathsf{def}_{\{z\}}\ x\langle u\rangle \mid y\langle\rangle \triangleright t\langle u\rangle \wedge t\langle u\rangle \mid z\langle v\rangle \triangleright P && \mathsf{in}\ x\langle 1\rangle \mid x\langle 2\rangle \mid y\langle\rangle. \end{aligned}$$

While these processes have the same traces, the latter can reduce to $\mathsf{def}_{\{z\}}\ x\langle u\rangle \mid y\langle\rangle \triangleright t\langle u\rangle \wedge t\langle u\rangle \mid z\langle v\rangle \triangleright P$ in $t\langle 1\rangle \mid x\langle 2\rangle$, and this internal step cannot be simulated by the former process. This well-known problem of *gradual commitment* is further discussed in [14], where coupled simulations are used to relate the two processes above.

6. The discriminating power of matching

In this section, we relate weak bisimulation to the standard *barbed equivalence* semantics. As in other process calculi, their coincidence relies on the discriminating power of an additional, *name matching* operator [22, 4].

6.1. Barbed congruence

The main semantics of the join-calculus in [15] is *barbed congruence*. This equivalence was initially studied for the π-calculus as a "reduction-based equivalence" in [29]. We recall its definition below, and refer to the mentioned papers for discussion.

Definition 26. The *output barb* $\Downarrow_x$ is a predicate over open RCHAMS that tests for the potential emission of messages: $\mathscr{S} \Downarrow_x$ when $\mathscr{S} \Rightarrow C[x\langle\tilde{v}\rangle]$ for some evaluation context $C[\,\cdot\,]$ such that x is free.

A relation ϕ is a *barbed simulation* if, whenever $\mathscr{S}\ \phi\ \mathscr{T}$, we have

(1) $\mathrm{xv}(\mathscr{S}) = \mathrm{xv}(\mathscr{T})$;
(2) if $\mathscr{S} \Downarrow_x$ then $\mathscr{T} \Downarrow_x$;
(3) if $\mathscr{S} \equiv \rightarrow \equiv \mathscr{S}'$ then $\mathscr{T}'$ such that $\mathscr{T} \Rightarrow \mathscr{T}'$ and $\mathscr{S}'\ \phi\ \mathscr{T}'$.

A relation ϕ is a *barbed bisimulation* if both ϕ and ϕ^{-1} are barbed simulations. *Barbed congruence* $\approx_{\mathrm{b}}$ is the largest barbed bisimulation that is preserved by application of open join-calculus contexts.

The first clause departs from the standard definition of barbed bisimulation [29]; it demands that bisimilar processes have the same extruded names. For instance, we have $\mathsf{def}_{\{x\}}\ x\langle u\rangle \triangleright \mathsf{0}\ \mathsf{in}\ \mathsf{0} \not\approx_{\mathrm{b}} \mathsf{0}$ by definition.

6.2. Weak bisimilarity versus barbed congruence

By definition of rule EXT, we have $\mathscr{S} \Downarrow_x$ if and only if $\mathscr{S} \Rightarrow \xrightarrow{S\tilde{x}\langle\tilde{v}\rangle} \mathscr{S}'$. Weak bisimilarity is thus a barbed bisimulation, and also a congruence (Theorem 12), hence it is finer than barbed congruence ($\approx \subset \approx_{\mathrm{b}}$). This containment is strict, as can be seen from the paradigmatic example of barbed congruence:

$$x\langle z\rangle \approx_{\mathrm{b}} \mathsf{def}\ u\langle v\rangle \triangleright z\langle v\rangle \mathsf{in}\ x\langle u\rangle.$$

That is, emitting a free name z is the same as emitting a bound name u that forwards all the messages it receives to z, because the extra internal move for every use of u is not observable. On the contrary, weak bisimilarity separates these two processes because their respective extrusion labels reveal that z is free and u is extruded. Since the contexts of the open join-calculus cannot identify names in messages, more powerful contexts are required to reconcile the two semantics.

6.3. Name matching

In this section only, we extend the syntax of the join-calculus with a name matching operator, in the same style as [28]

$$A \stackrel{\mathrm{def}}{=} \ldots \mid [x = y]A \quad P \stackrel{\mathrm{def}}{=} \ldots \mid [x = y]P.$$

Accordingly, we extend our chemical machines with a new reduction rule.

$$\textsc{Match} \quad \vdash_S [x = x]A \rightarrow \vdash A.$$

A technical drawback of name matching is that global renamings do not preserve weak bisimilarity anymore. For instance, $\mathsf{0} \approx [x = y]\, x\langle\rangle$, while after applying the renaming $\{^x/_y\}$, $\mathsf{0} \not\approx [x = x]\, x\langle\rangle$. Accordingly, weak bisimilarity is not a congruence anymore. For instance, the context $C[\,\cdot\,] \stackrel{\mathrm{def}}{=} \mathsf{def}\ z\langle x, y\rangle \triangleright [\,\cdot\,]\ \mathsf{in}\ z\langle u, u\rangle$ separates $\mathsf{0}$ and $[x = y]\, x\langle\rangle$.

In order to retain the full congruence property, we may consider the coarsest equivalence contained into $\approx$ and preserved by global renaming. It is possible to prove, in the same style as for Theorem 12, that this equivalence is indeed a congruence; however this equivalence is not a bisimulation.

Alternatively, we consider equivalences that are not full congruences. The next lemma restates the partial congruence property of Lemma 13 in the presence of matching – the same proof techniques apply unchanged.

Lemma 27. *Weak bisimilarity is preserved by application of evaluation contexts with name matching.*

We adapt our definition of barbed congruence accordingly:

Definition 28. *Barbed equivalence* $\approx_{\mathrm{be}}$ is the largest barbed bisimulation in the open join-calculus with matching that is preserved by application of all evaluation contexts of the plain join-calculus with matching.

Barbed equivalence now separates $x\langle z\rangle$ from $\mathsf{def}\ u\langle v\rangle \triangleright z\langle v\rangle\ \mathsf{in}\ x\langle u\rangle$ by using the context $\mathsf{def}\ x\langle y\rangle \triangleright [y=z]\, a\langle\rangle\ \mathsf{in}\ [\,\cdot\,]$, and weak bisimilarity clearly remains finer than barbed equivalence. In the next section, we focus on the converse property. (As a corollary of Theorem 29, it turns out that barbed equivalence is also preserved by application of evaluation contexts with extruded names.)

6.4. Barbed equivalence is a weak bisimulation

In a process calculus with matching, the coincidence of weak bisimilarity and barbed equivalence is not easy to prove or disprove. A first positive result is given in [22] for the ν-calculus. In the π-calculus, the question was raised in [29], and closed with both positive and negative results depending on slightly different definitions of equivalences [16]. Sangiorgi [33] proves that standard early bisimulation coincides with barbed equivalence both in CCS and in the monadic π-calculus. The technique consists of building contexts that test all possible behaviors of a process under bisimulation. This technique requires infinite contexts with infinite numbers of free names and of recursive constants. Such contexts are otherwise never considered in congruence properties, and cannot be expressed using constructs such as replication instead of parameterized recursive constants. In recent works, partial results for variants of the π-calculus are obtained by using the same technique; since only finite contexts are available, the coincidence is established only for image-finite processes (i.e., processes with a finite set of derivatives).

As discussed in [16, 14], there are actually *two* ways of defining barbed equivalence. In "classical" barbed equivalences, a congruence property is required once, before checking that the two processes are bisimilar. In Definitions 26 and 28, however, as in the ν-calculus [22], congruence and bisimulation properties are required at the same time. This technical choice is essential to obtain a simple proof that weak bisimulation and barbed equivalence coincide: instead of considering whole synchronization trees, we can focus on single transitions. For every labeled transition, we apply a specific evaluation context that "captures" the label, then disappears up to barbed equivalence. (The corresponding π-calculus proof appears in [16]; further discussion of barbed equivalences, including a proof that the two definitions of barbed equivalence yield the same equivalence appears in [14].)

Theorem 29. *With name matching, we have* $\approx_{\mathrm{be}} = \approx$.

In the following proof, we focus on the inclusion $\approx_{\mathrm{be}} \subseteq \approx$; the converse inclusion holds by Definition 28 and Lemma 27. The problematic case is extrusion, because a context of the join-calculus must define a name in order to detect an output transition on that name; this case is handled by creating a permanent relay for all other messages on that name. Without additional care, this relay can be detected by name matching, so we use instead a family of contexts that separate two aspects of a name. For every name $x \in \mathcal{N}$, we let

$$R_x[\cdot] \stackrel{\text{def}}{=} \mathsf{def}\ x\langle \tilde{y}\rangle \triangleright x'\langle \tilde{y}\rangle \mathsf{in}\ v_x\langle x\rangle \mid [\cdot],$$

where the length of $\tilde{y}$ matches the arity of x. Assuming $x \in \mathrm{fv}(A)$, the process $R_x[A]$ uses x' as a free name instead of x, and forwards all messages from x to x'. The context should still be able to discriminate whether the process sends the name x or not by using name matching; this extra capability is supplied in an auxiliary message $v_x\langle x\rangle$. The next proposition describes reductions within contexts $R_x[\,\cdot\,]$.

Proposition 30. *If $R_x[A] \Rightarrow \mathscr{T}$ then for some A' we have $A \Rightarrow A'$ and $R_x[A'] \approx \mathscr{T}$.*

Proof. We prove by induction on n that if $R_x[A]\ (\equiv \rightarrow \equiv)^n \approx \mathscr{T}$ then $A \Rightarrow A'$ and $R_x[A'] \approx \mathscr{T}$. The base case is immediate; we distinguish two situations for the inductive case, according to the initial reduction step.

- If this step occurs in A, we apply the induction hypothesis.
- Otherwise, this step uses the single rule of the context; it consumes a message $x\langle \tilde{w}\rangle$ and replaces it with a message $x'\langle \tilde{w}\rangle$; this step commutes with all subsequent steps and preserves weak bisimulation (Example 21). We apply the induction hypothesis for all other steps and substitute weak bisimulation for the last step. □

Informally, the contexts $R_x[\,\cdot\,]$ are the residuals of contexts that test for labels of the form $\{x\}\bar{y}\langle x\rangle$. The next lemma captures the essential property of $R_x[\,\cdot\,]$.

Lemma 31 (Accommodating the extrusions). *For all open processes A and B such that $x \in \mathrm{xv}(A) \cup \mathrm{xv}(B)$ and x', v_x are not in the interface of A and B, we have $A \approx_{\mathrm{be}} B$ if and only if $R_x[A] \approx_{\mathrm{be}} R_x[B]$.*

Proof. If $A \approx_{\mathrm{be}} B$, then the closure property of $\approx_{\mathrm{be}}$ with respect to plain evaluation contexts yields $R_x[A] \approx_{\mathrm{be}} R_x[B]$. To prove the converse implication, we let

$$\phi \stackrel{\text{def}}{=} \left\{ (A,B) \;\middle|\; \begin{array}{l} x, \notin \mathrm{xv}(A) \cup \mathrm{xv}(B) \\ R_x[A] \approx_{\mathrm{be}} R_x[B] \text{ for some fresh names } x' \text{ and } v_x \end{array} \right\}$$

and establish that ϕ is a barbed equivalence. Let $A\ \phi\ B$.

(1) ϕ is a weak bisimulation for silent steps. If $A \equiv \rightarrow \equiv A'$, then $R_x[A] \equiv \rightarrow \equiv R_x[A']$. By hypothesis $R_x[A] \approx_{\mathrm{be}} R_x[B]$ hence this reduction is simulated by reductions $R_x[B] \Rightarrow \mathscr{T}$ with $R_x[A'] \approx_{\mathrm{be}} \mathscr{T}$. By Proposition 30, we have $B \Rightarrow B'$ and $R_x[B'] \approx_{\mathrm{be}} \mathscr{T}$. By transitivity $R_x[A'] \approx_{\mathrm{be}} R_x[B']$ and thus $A'\ \phi\ B'$.

(2) ϕ respects the barbs. If $A \Downarrow_y$ with $y \neq x$, then also $R_x[A] \Downarrow_y$ by using the same reductions; by hypothesis $R_x[B] \Downarrow_y$ and, since $y \notin \{v_x, x'\}$ we must have $B \Downarrow_y$. Similarly, if $A \Downarrow_x$ then $R_x[A] \Downarrow_{x'}$, $R_x[B] \Downarrow_{x'}$, and $B \Downarrow_x$.

(3) ϕ is preserved by application of evaluation contexts of the plain join-calculus with matching. It suffices to show the closure property for any context of the form $C[\,\cdot\,] = \mathsf{def}\ \bigwedge_{i=1}^{n} J_i \triangleright P_i\ \mathsf{in}\ Q \,|\, [\,\cdot\,]$. We show that $R_x[C[A]] \approx_{\mathrm{be}} R_x[C[B]]$ by translating $C[\,\cdot\,]$ to another context $[\![C]\!][\,\cdot\,]$ that binds v_x, receives x on v_x, uses x everywhere except for the definition of x', and re-applies R_x on the outside. We pick fresh names ρ and w_x, and distinguish two cases according to the scope of x, In case $C[\,\cdot\,]$ binds x ($x \in \bigcup_{i=1}^{n} \mathrm{dv}(J_i)$), we use the translation

$$[\![C]\!][\cdot] \stackrel{\mathrm{def}}{=} R_x[\mathbf{0}] \,|\, \mathsf{def} \bigwedge_{i=1}^{n} J_i\{x'/x\} \,|\, \rho\langle x\rangle \triangleright P_i \,|\, \rho\langle x\rangle \wedge v_x\langle x\rangle \triangleright \rho\langle x\rangle \,|\, Q\ \mathsf{in}\ [\cdot].$$

Otherwise, we use the translation

$$[\![C]\!][\cdot] \stackrel{\mathrm{def}}{=} \mathsf{def} \bigwedge_{i=1}^{n} J_i \,|\, \rho\langle x\rangle \triangleright P_i \,|\, \rho\langle x\rangle \wedge v_x\langle x\rangle \triangleright \rho\langle x\rangle \,|\, w_x\langle x\rangle \,|\, Q\ \mathsf{in}\ [\cdot].$$

In each case, we establish $[\![C]\!][R_x[A]] \approx R_x[C[A]]$ by a standard bisimulation argument. We then use the inclusion $\approx \subseteq \approx_{\mathrm{be}}$, apply the congruence property for $[\![C]\!][\,\cdot\,]$ to the hypothesis $R_x[A] \approx_{\mathrm{be}} R_x[B]$, and obtain $C[A]\ \phi\ C[B]$ by transitivity. □

Proof of Theorem 29. We establish that $\approx_{\mathrm{be}}$ is a weak bisimulation up to structural equivalence. Let $A \approx_{\mathrm{be}} B$. For each kind of transition $A \equiv \xrightarrow{\alpha} \equiv A'$ we use a context that specifically consumes this transition, then behaves as the trivial context $[\,\cdot\,]$.

Internal step $\equiv \rightarrow \equiv$. This follows from the bisimulation property of $\approx_{\mathrm{be}}$.

Intrusion $\equiv \xrightarrow{x\langle \tilde{y}\rangle} \equiv$. Independently of the values $\tilde{y}$, intrusion is enabled on x if and only if $x \in \mathrm{xv}(A)$, and we have both $A \equiv \xrightarrow{x\langle \tilde{y}\rangle} \equiv A' \equiv A \,|\, x\langle \tilde{y}\rangle$ and $B \xrightarrow{x\langle \tilde{y}\rangle} B \,|\, x\langle \tilde{y}\rangle$. We apply the congruence property of $\approx_{\mathrm{be}}$ for the context $[\,\cdot\,] \,|\, x\langle \tilde{y}\rangle$.

Extrusion $\equiv \xrightarrow{S\tilde{x}\langle y_1,\ldots,y_n\rangle} \equiv$. Let $m \in 0 \ldots n$ be the cardinal of S. Without loss of generality, we assume that the freshly extruded names in S are the first arguments of the message ($S = \{y_1, \ldots, y_m\}$). We also assume that $S \cap (\mathrm{fv}(A) \cup \mathrm{fv}(B)) = \emptyset$. We use the congruence property for the context

$$\begin{aligned} E[\cdot] \stackrel{\mathrm{def}}{=} \mathsf{def}\ & x\langle \tilde{z}\rangle \,|\, grabs\langle\rangle \triangleright V \\ & \wedge\ x\langle \tilde{z}\rangle \,|\, done\langle\rangle \triangleright x'\langle \tilde{z}\rangle \,|\, done\langle\rangle \\ & \wedge\ once\langle\rangle \triangleright test\langle\rangle \\ & \wedge\ once\langle\rangle \,|\, done\langle\rangle \triangleright done\langle\rangle \\ & \mathsf{in}\ once\langle\rangle \,|\, grab\langle\rangle \,|\, v_x\langle x\rangle \,|\, [\cdot] \end{aligned}$$

with the auxiliary definitions

$$T \stackrel{\text{def}}{=} \{(y_i, z_i) \mid m < i \leqslant n\} \cup \{(z_i, z_j) \mid 0 \leqslant i \leqslant m < j \text{ and } y_i = y_j\},$$

$$F \stackrel{\text{def}}{=} (\text{fv}(A) \cup \text{fv}(B) \cup \text{xv}(A) \cup xv(B)) \times \{z_1, \ldots, z_m\} \cup \{(z_i, z_j) \mid 0 \leqslant i, j \leqslant m \text{ and } y_i \neq y_j\},$$

$$V \stackrel{\text{def}}{=} \bigcap_{(y,z) \in T} [y = z]done\langle\rangle \mid \prod_{(y,z) \in F} [y = z]test\langle\rangle,$$

where the names $\tilde{v}$, *grab*, *once*, *done*, *test*, and v_x are fresh, and where the notation $\bigcap$ in V abbreviates a sequence of tests. Informally, $E[\,\cdot\,]$ receives a message on x, checks that its arguments match the expected label using the process V, then behaves like $R_x[\,\cdot\,]$. The parameters are bound to names $\tilde{v}$, then compared to the names expected on the label. The finite set $T \subset \mathcal{N} \times \mathcal{N}$ gathers all pairs of names that must coincide: previously known names and repeated fresh names. The finite set $F \subset \mathcal{N} \times \mathcal{N}$ gathers all pairs of names that must be distinct: freshly extruded names are distinct from any previously visible name, and pairwise distinct unless syntactically the same on the label.

- if $A \equiv \xrightarrow{S\tilde{x}\langle y_1, \ldots, y_n\rangle} \equiv A'$, then $E[A] \Rightarrow \approx R_x[A']$.
  Let t be the cardinal of T. We use the series of $t + 2$ reductions that consumes $x\langle\tilde{y}\rangle \mid grab\langle\rangle$, passes the series of positive tests T in V – thus releasing the message $done\langle\rangle$ – then consumes $once\langle\rangle \mid done\langle\rangle$ – thus removing the barb $\Downarrow_{test}$. Using structural rearrangement, the remains of the context can be written as

$$E'[\cdot] \stackrel{\text{def}}{=} \textsf{def } x\langle\tilde{z}\rangle \mid done\langle\rangle \triangleright x'\langle\tilde{z}\rangle \mid done\langle\rangle \wedge D \textsf{ in } done\langle\rangle \mid v_x\langle x\rangle \mid [\cdot] \mid \prod_{(y,z) \in F} [y = z]test\ \langle\rangle,$$

  where D and $\prod_{(y,z) \in F} [y = z]test\langle\rangle$ are inert. We easily establish that $E'[A'] \approx R_x[A']$ for any A' where the names *grab*, *once*, *done*, and v_x are fresh.
- if $E[B] \Rightarrow U$ and $U \not\Downarrow_{test}$, then $B \Rightarrow \xrightarrow{S\tilde{x}\langle y_1, \ldots, y_n\rangle} \equiv B'$ with $R_x[B'] \approx_{\text{be}} U$. In order to get rid of the barb $\Downarrow_{test}$, every step in the series detailed above is required to emit the message $done\langle\rangle$. Let $x\langle\tilde{z}\rangle$ be the first message received by $E[\,\cdot\,]$. We decompose the reductions leading to U as follows. Before the first reception on x, all reductions are internal to B; after this reception, all reductions are either internal steps in the derivative of B, internal steps to $E[\,\cdot\,]$ exactly as described above, or further receptions on x in $E[\,\cdot\,]$, which do not affect barbed equivalence.

  Since the barb $\Downarrow_{test}$ disappears, every comparison in the series on the first line of V has succeeded, which ensures $y_i = z_i$ for $i = m+1 \ldots n$. Besides, no comparison in the parallel composition on the second line of V may succeed, as this would reintroduce a $test\langle\rangle$ message, hence the names $z_1, \ldots, z_m$ are all fresh names and we can perform α-conversion before the reduction to enforce $y_i = z_i$ for $i = 1 \ldots m$.

We write $B \Rightarrow C[x\langle\tilde{y}\rangle]$ for the series of reductions internal to B, we choose B' such that $C[x\langle\tilde{y}\rangle] \equiv \xrightarrow{S\tilde{x}\langle y_1,\ldots,y_n\rangle} \equiv B'$, and we use the first item above to obtain $R_x[B'] \approx_{\mathrm{be}} U$.

Let us assume $A \approx_{\mathrm{be}} B$ and $A \equiv \xrightarrow{S\tilde{x}\langle y_1,\ldots,y_n\rangle} \equiv A'$. By context closure property, $E[A] \approx_{\mathrm{be}} E[B]$. By barbed equivalence, the reductions $E[A] \Rightarrow E'[A']$ must be simulated by some reductions $E[B] \Rightarrow U$ with $E'[A'] \approx_{\mathrm{be}} U$. Since $E'[A] \not\Downarrow_{test}$, we have $U \not\Downarrow_{test}$ and thus $R_x[A'] \approx_{\mathrm{be}} R_x[B']$. By Lemma 31, this entails $A' \approx_{\mathrm{be}} B'$. □

7. A comparison with the π-calculus

The join-calculus can be considered as a disciplined version of the asynchronous π-calculus, in which the syntax enforces locality in the usage of names. These calculi have a lot in common, and our semantics largely draw upon the bisimulations developed for the π-calculus [28, 29, 36, 21, 4]. In this section, we relate our definitions to previous proposals in the literature and we compare the equivalences obtained by applying similar definitions to both calculi.

In the sequel, we focus on the asynchronous π-calculus with the following grammar for processes:

$$P ::= \mathbf{0} \mid \bar{x}\langle v\rangle \mid x(y).P \mid \nu x.P \mid !P \mid P \mid P.$$

We refer to Boudol [11] for the operational semantics. In short, the basic reduction step matches complementary pairs of emission and reception $(\bar{x}\langle v\rangle \mid x(y).Q \to Q\{^{v}/_{y}\})$; other transitions render intrusion or extrusion of messages with labels that carry the same information as those of the open RCHAM. Crucially, output labels explicitly mention the names being extruded.

The join-calculus and the asynchronous π-calculus have the same expressive power, at least up to barbed congruence [15], but their treatment of names is rather different. In the open join-calculus, the interface of a process consists of two disjoint sets of names; free names are used exclusively for extrusions; extruded names have local definitions and can be used in silent steps.

In the π-calculus, on the contrary, the same free name can be used for intrusion, extrusion, and internal reduction. Besides, a received name can be used to set up new receivers, as in $x(y).y(z).P$, thus extending the discriminating power of contexts. (This would amount to redefining the name in the join-calculus.) To illustrate this point, we consider a rough π-calculus encodings of the processes mentioned in the discussion of barbed congruence in Section 6.2; the processes $\bar{x}\langle z\rangle$ and $\nu u.(!u(v).\bar{z}\langle v\rangle \mid \bar{x}\langle u\rangle)$ are not barbed congruent, and this can be established using a context that invalidates locality. For example, if we choose the context $C[\cdot] \stackrel{\mathrm{def}}{=} \nu x,z.(x(a).a(u).\bar{u}\langle\rangle \mid \bar{z}\langle b\rangle \mid [\cdot])$, we obtain $C[\bar{x}\langle z\rangle] \Downarrow_b$ and $C[\nu u.(!u(v).\bar{z}\langle v\rangle \mid \bar{x}\langle u\rangle)] \not\Downarrow_b$.

We now compare the labeled semantics for the open join-calculus with previous proposals for the asynchronous π-calculus, notably [21, 4]. In these works, a major issue is to adapt the semantics inherited from the original π-calculus to asynchrony: since message output is not a prefix anymore, emitters in the context cannot detect whether their messages are actually read. Technically, this leads to a special treatment of input actions, either in the definition of transitions or in the definition of bisimulation.

In [21], Honda and Tokoro take as observational semantics the standard notion of weak bisimulation. As a consequence, they have to change the definition of transitions. Take for instance the π-calculus processes $\mathsf{0}$ and $x(u).\bar{x}\langle u\rangle$. In order to progress, the process $x(u).\bar{x}\langle u\rangle$ has to consume an emission $\bar{x}\langle v\rangle$, thus exhibiting a new emission $\bar{x}\langle v\rangle$. Similarly, the process $\mathsf{0}$ "takes" $\bar{x}\langle v\rangle$ and "gives" the same $\bar{x}\langle v\rangle$ (just nothing is consumed or produced). Therefore $\mathsf{0}$ and $x(u).\bar{x}\langle u\rangle$ should be considered equal in an asynchronous setting: indeed, they are barbed congruent. However, standard bisimulation obviously discriminates between $\mathsf{0}$ and $x(u).\bar{x}\langle u\rangle$. To alleviate this problem, Honda and Tokoro adopt an operational model where asynchrony of communication is rendered as the total receptiveness of the process. In other words, any message can be intruded from the environment at any time, using extended structural equivalence in combination with the rule

$$\textsc{Input} \quad \mathsf{0} \xrightarrow{x\langle v\rangle} \bar{x}\langle v\rangle.$$

The intrusion rule of the open RCHAM is reminiscent of this kind of operational semantics, with two important differences: (i) our rule INT can be used only with previously extruded names; conversely, their input rule immediately yields an infinite transition system; and (ii) extraneous inputs in the join-calculus are not observable, and thus messages not used by the process can safely be discarded.

In [4], Amadio et al. take the opposite approach. They keep the standard synchronous semantics and they modify the notion of bisimulation. For two processes to be bisimilar, they do not require that every input be simulated by an input of the other side. Rather, they supplement bisimulation with a *delay clause* that can simulate an input on one side by adding a new message in parallel on the other side. Their resulting asynchronous bisimulation offers three advantages: it eliminates the need for total receptiveness, it is consistent with external sum, and it relies on a widely used semantics.

The asynchronous bisimulation of Section 4 relies on similar motivations – handling asynchrony in the definition of bisimulation with a relaxed clause for intrusions – but our intrusion clause is different from theirs: we deal with multiple intrusions, and we allow the simulating process to perform arbitrary internal moves after parallel composition with the intruded messages; in their terminology, this would place our equivalence between 1-bisimulation and asynchronous bisimulation. Also, the JOPEN RCHAM equipped with asynchronous bisimulation is not meant to be a main semantics, but rather a technical device to reduce branching in the underlying transition system; arguably, the open RCHAM gives a more intuitive meaning to extruded names. We

complete this comparison by pointing out two technical differences:

- The intrusion clause for their asynchronous bisimulation does not suitably carry over to the join-calculus. In our syntax, this clause is

$$\text{if } P \equiv \stackrel{x\langle\tilde{u}\rangle}{\rightarrow} \equiv P' \text{ then } \begin{array}{l} \text{either } Q \Rightarrow \stackrel{x\langle\tilde{u}\rangle}{\rightarrow} \Rightarrow Q' \text{ and } P'\phi Q' \\ \text{or } Q \Rightarrow Q' \text{ and } P'\phi\ (Q' \mid x\langle\tilde{u}\rangle). \end{array}$$

  Its generalization to the JOPEN RCHAM obtained by allowing join-intrusions leads to a notion of equivalence strictly finer than weak bisimilarity. Take, for instance, the following equivalence:

$$\underset{\{x,y\}}{\mathsf{def}}\ x\langle z\rangle \mid y\langle\rangle \rhd z\langle\rangle \ \mathsf{in}\ x\langle a\rangle$$
$$\underset{\{x,y\}}{\mathsf{def}}\ x\langle z\rangle \mid y'\langle\rangle \rhd z\langle\rangle \wedge y\langle\rangle \rhd y'\langle\rangle \ \mathsf{in}\ x\langle a\rangle.$$

  The two processes are weakly bisimilar (see Example 24), but the first process can perform a double intrusion $x\langle b\rangle \mid y\langle\rangle$ that emits the message $b\langle\rangle$, while the second process can neither perform the same intrusion nor exclude the emission of $a\langle\rangle$ instead of $b\langle\rangle$.
- Weak bisimulation and asynchronous bisimulation coincide in the open join-calculus, for simple reasons. In contrast, the correspondence between the bisimulations of Honda and Tokoro [21] and those of Amadio et al. [4], discussed in details in the latter paper, is a delicate issue; it is unclear, for instance, whether both approaches yield the same relation in the weak case.

Other devices have been proposed to reduce the number of transitions to consider when comparing π-calculus processes. An alternative approach is to use typed interfaces and typed bisimulations in order to structure interaction with the environment [9]. A more dynamic approach is to prune families of extraneous transitions from the synchronization tree. In [30] for instance, only transitions on "active names" are considered.

Independently, several notions of locality have been considered in a pure π-calculus setting. An early reference is [20], where an object calculus is derived from the asynchronous π-calculus. To maintain the integrity of objects, processes that receive object names cannot receive on those names. In [3], the π-calculus is equipped with a type system that bans multiple receptors for the same message. It is proved that this π-calculus with unique receptors is expressive enough to simulate the asynchronous π-calculus. In [6], a fragment of the π-calculus is considered where received names cannot be used for input (i.e. cannot be redefined) and this language is compositionally compiled into a simple substitution-free calculus. The condition of transmitting output capabilities is further strengthened into *uniform receptiveness* in [35] by also demanding that replicated name definitions be available as soon as the name is created. Uniform receptiveness is actually a locality property, which is formulated by syntactic means in join-calculus and by means of a type system in the π-calculus.

8. Conclusions

In this paper, we have developed a theory of bisimulations for the join-calculus. Starting from the reduction-based definitions of Fournet and Gonthier [15], we extended the RCHAM with intrusions and extrusions, in two different styles, and we studied for each style a relevant definition of labeled bisimulation. Asynchronous bisimulations seem best suited for proofs, as they relate smaller synchronization trees, but both labeled equivalences should be amenable to automated verification by means of the existing algorithms (see, for instance, [37, 30]).

The precise role of name matching deserves further investigation. The coincidence of weak bisimilarity and barbed equivalence holds only in the presence of matching, which invalidates useful process equalities. It would be interesting to find direct, purely co-inductive characterizations of barbed equivalence in the absence of matching. This issue is addressed in [9] in a π-calculus setting, and in [25] in an asynchronous calculus closely related to the join-calculus. In this latter work, Merro and Sangiorgi introduce a new extrusion clause requiring that either the extruded name be the same on both sides of the bisimulation or that the two resulting processes can be made indistinguishable by adding a relaying process on the fly. They prove that this coarser bisimulation coincides with barbed equivalence for image-finite processes.

Our labeled semantics can be carried over to richer variants of the join-calculus that account for distribution and agent mobility [17, 12], or for security properties [1, 2]. For example, an open variant of the join-calculus is used in [2] to express authentication properties; in that setting, extruded names appearing in a parallel composition of processes are interpreted as mutually-trusted channels between those processes. Besides, the information conveyed by the labels can also be supplemented with other properties of interest. Such approaches have been recently explored in other process calculi by Amadio [3] and Hennessy and Riely [32] for coping with partial failures in distributed systems, and by Boreale et al. [8] for dealing with cryptography.

Appendix: Additional proofs

Proof of Lemma 11. In each case, we exhibit a bisimulation that contains ϕ:

(1) The definition of weak bisimulation already takes into account structural rearrangements, so $\equiv\phi\equiv$ is clearly a weak bisimulation.

(2) The relation $\approx\phi\approx$ is a weak bisimulation; this directly follows from the transitivity of $\approx$. (However, the lemma would not hold if we used $\approx\phi\approx$ instead of $\phi\approx$ in the simulation clause.)

(3) We prove that the relation

$$\psi \stackrel{\text{def}}{=} \{(\mathscr{S}\backslash S, \mathscr{T}\backslash S) \mid \mathscr{S}\phi\mathscr{T} \text{ and both restrictions are defined}\}$$

is a weak bisimulation up to structural equivalence. All intrusions and internal reductions that are enabled after applying the restriction are also enabled before;

and additional internal steps $\equiv \rightarrow \equiv$ are preserved by the restriction, so we can obtain the required simulation properties by reporting series of transitions obtained by simulation before restriction.

We consider the case of extrusions: assume $\mathscr{S} \phi \mathscr{T}$ and $\mathscr{S}\backslash S \equiv \xrightarrow{S'\tilde{x}\langle\tilde{v}\rangle} \mathscr{S}'$. Taking $T = S \cap S'$, we rewrite the transition as $\mathscr{S}\backslash(U \uplus T) \equiv \xrightarrow{(V \uplus T)\tilde{x}\langle\tilde{v}\rangle} \mathscr{S}'$ then argue that, for some solution $\mathscr{S}''$, we have $\mathscr{S} \equiv \xrightarrow{V\tilde{x}\langle\tilde{v}\rangle} \mathscr{S}''$ and $\mathscr{S}' \equiv \mathscr{S}''\backslash U$. By simulation hypothesis, there is a solution $\mathscr{T}''$ such that $\mathscr{T} \Rightarrow \xrightarrow{V\tilde{x}\langle\tilde{v}\rangle} \Rightarrow \mathscr{T}''$ and $\mathscr{S}'' \phi \mathscr{T}''$, so $\mathscr{T}\backslash(U \uplus T) \Rightarrow \xrightarrow{(V \uplus T)\tilde{x}\langle\tilde{v}\rangle} \Rightarrow \mathscr{T}''\backslash U$. We finally obtain $\mathscr{S}' \psi \mathscr{T}''\backslash V$. □

Proof of Lemma 13. We show that the relation $\phi \stackrel{\text{def}}{=} \{(C[A], C[B]) \mid A \approx B\}$ is a weak simulation up to restriction and structural rearrangement.

Transitions that do not depend on the process placed in the context $C[\cdot]$ are in direct correspondence on both sides of ϕ, and yield new pairs of related processes. Likewise, transitions in A that are not affected by the context $C[\cdot]$ are simulated in B by hypothesis, yielding new related processes. This leaves us with two cases:

- *Some messages of $C[\cdot]$ are consumed by a reaction rule of A.* Let $x_1\langle\tilde{v}_1\rangle \mid \ldots \mid x_n\langle\tilde{v}_n\rangle$ be these messages. We must have $x_1, \ldots, x_n \in \mathrm{xv}(A)$ and, by applying Proposition 6, the internal move $C[A] \equiv \rightarrow \equiv E$ can be displayed as

$$\begin{aligned} A &\equiv \xrightarrow{x_1\langle\tilde{v}_1\rangle} \cdots \xrightarrow{x_n\langle\tilde{v}_n\rangle} \equiv \rightarrow \equiv A' \\ C[\cdot] &\equiv C'[x_1\langle\tilde{v}_1\rangle \mid \ldots \mid x_n\langle\tilde{v}_n\rangle \mid \cdot], \end{aligned}$$

  where the latter relation abbreviates a structural equivalence that holds for any process with the same extruded names as A substituted for $[\cdot]$, and where $E \equiv C'[A']$. Since $A \approx B$, the first series of transitions can be simulated by B as $B \Rightarrow \xrightarrow{x_1\langle\tilde{v}_1\rangle} \cdots \Rightarrow \xrightarrow{x_n\langle\tilde{v}_n\rangle} \Rightarrow B'$ with $A' \approx B'$. By applying Proposition 6 again, we obtain $C[B] \Rightarrow C'[B']$, and $C'[A'] \phi C'[B']$.
- *Some messages of A are consumed by a reaction rule of $C[\cdot]$.* Let $x_1\langle\tilde{v}_1\rangle \mid \ldots \mid x_n\langle\tilde{v}_n\rangle$ be these messages. We have $x_1, \ldots, x_n \in \mathrm{fv}(A)$ and, by applying Proposition 6, the internal move $C[A] \equiv \rightarrow \equiv E$ can be displayed as

$$\begin{aligned} & & A &\equiv \xrightarrow{S_1\tilde{x}_1\langle\tilde{v}_1\rangle} \cdots \xrightarrow{S_n\tilde{x}_n\langle\tilde{v}_n\rangle} \equiv A' \\ C[x_1\langle\tilde{v}_1\rangle \mid \ldots \mid x_n\langle\tilde{v}_n\rangle \mid \cdot], & & \equiv \rightarrow \equiv & & C'[\cdot], \end{aligned}$$

  where the second series applies for all processes within the context that extrude every name in $S \stackrel{\text{def}}{=} \bigcup_{i=1}^{n} S_i$, and where $E = C'[A']\backslash S$. Since $A \approx B$, the first series can be simulated by B as

$$B \Rightarrow \xrightarrow{S_1\tilde{x}_1\langle\tilde{v}_1\rangle} \cdots \Rightarrow \xrightarrow{S_n\tilde{x}_n\langle\tilde{v}_n\rangle} \Rightarrow B'$$

  with $A' \approx B'$. By applying Proposition 6 we can defer the extrusions, perform all the internal steps of B within the context $C[\cdot]$, then perform the internal step from

$C[\cdot]$ to $C'[\cdot]$. We obtain $C[B] \Rightarrow C'[B']\backslash S$ and again we have related processes up to restriction. □

Proof of Lemma 14. We prove that the closure of $\approx$ for all valid global renamings

$$\phi \stackrel{\text{def}}{=} \{(\mathscr{S}\sigma, \mathscr{T}\sigma) \mid \mathscr{S} \approx \mathscr{T} \text{ and } \sigma \text{ global renaming for } \mathscr{S} \text{ and } \mathscr{T}\}$$

is a weak bisimulation. The proof relies on the analysis of the effects of a global renaming on every chemical rule, as described in Lemma 7. Let $\mathscr{S}\sigma \,\phi\, \mathscr{T}\sigma$.

Intrusions $\mathscr{S}\sigma \equiv \stackrel{x\langle\tilde{v}\rangle}{\longrightarrow} \equiv \mathscr{S}_1$: By Lemma 7(3, second part), we have $\mathscr{S} \equiv \stackrel{\alpha}{\rightarrow} \equiv \mathscr{S}'$ for some α and $\mathscr{S}'$ such that $\alpha\sigma = x\langle\tilde{v}\rangle$ and $\mathscr{S}'\sigma = \mathscr{S}_1$. Since $\mathscr{S} \approx \mathscr{T}$, we have $\mathscr{T} \Rightarrow \stackrel{\alpha}{\rightarrow} \Rightarrow \mathscr{T}''\sigma'$ for some σ' such that $\mathscr{S}' = \mathscr{S}''\sigma'$ and $\mathscr{S}'' \approx \mathscr{T}''$. By Lemma 7(3, first part), there is a transition after renaming $\mathscr{T}\sigma \Rightarrow \stackrel{x\langle\tilde{v}\rangle}{\longrightarrow} \Rightarrow (\mathscr{T}''\sigma')\sigma$, and the pair $\mathscr{S}_1 = (\mathscr{S}''\sigma')\sigma \;\phi\; (\mathscr{T}''\sigma')\sigma$ meets the bisimulation clause for intrusion.

Extrusions $\mathscr{S}\sigma \equiv \stackrel{S\tilde{x}\langle\tilde{v}\rangle}{\longrightarrow} \equiv \mathscr{S}_1$: the proof is similar.

Internal move $\mathscr{S}\sigma \equiv \rightarrow \equiv \mathscr{S}_1$: by Lemma 7(2), there is a series of $n \geqslant 0$ extrusion-intrusion pairs followed by a silent move:

$$\mathscr{S} \equiv \stackrel{S_1\overline{x_1}\langle\tilde{v}_1\rangle}{\longrightarrow} \stackrel{y_1\langle\tilde{v}_1\rangle}{\longrightarrow} \cdots \stackrel{S_n\overline{x_n}\langle\tilde{v}_n\rangle}{\longrightarrow} \stackrel{y_n\langle\tilde{v}_n\rangle}{\longrightarrow} \equiv \rightarrow \equiv \mathscr{S}'$$

such that $y_i \in \mathrm{xv}(\mathscr{S})$ and $x_i\sigma = y_i\sigma$ for all $i \leqslant n$, and such that $\mathscr{S}'\backslash S \equiv \mathscr{S}_1$ with $S \stackrel{\text{def}}{=} \bigcup_{i=1\ldots n} S_i$. Since $\mathscr{S} \approx \mathscr{T}$, there is a matching series of transitions

$$\mathscr{T} \Rightarrow \stackrel{S_1\overline{x_1}\langle\tilde{v}_1\rangle}{\longrightarrow} \Rightarrow \stackrel{y_1\langle\tilde{v}_1\rangle}{\longrightarrow} \Rightarrow \cdots \Rightarrow \stackrel{S_n\overline{x_n}\langle\tilde{v}_n\rangle}{\longrightarrow} \Rightarrow \stackrel{y_n\langle\tilde{v}_n\rangle}{\longrightarrow} \Rightarrow \mathscr{T}'$$

such that $\mathscr{S}' \approx \mathscr{T}'$. Repeatedly applying Proposition 5, Lemma 8, and Lemma 7(2), we translate these transitions under the global renaming σ and obtain $\mathscr{T}\sigma \Rightarrow (\mathscr{T}'\backslash S)\sigma$. By Lemma 11(3), we have $\mathscr{S}'\backslash S \approx \mathscr{T}'\backslash S$, and finally $\mathscr{S}_1 \equiv (\mathscr{S}'\backslash S)\sigma \;\phi\; (\mathscr{T}'\backslash S)\sigma$. □

Proof of Theorem 12. We first establish that $\approx$ is preserved by application of contexts of the form $\mathsf{def}_S\ J \triangleright [\cdot]\ \mathsf{in}\ A$. Let us show that the relation

$$\phi \stackrel{\text{def}}{=} \{((\mathscr{D}, J \triangleright P \vdash_S \mathscr{A}), \mathscr{D}, J \triangleright Q \vdash_S \mathscr{A})) \mid P \approx Q\}$$

is a weak bisimulation up to weak bisimulation on the right. All transitions are in direct correspondence, except for reduction steps that consume messages in $\mathscr{A}$ to trigger the reaction rule $J \triangleright P$ or $J \triangleright Q$.

Let $\mathscr{S} \;\phi\; \mathscr{T}$, let $\mathscr{S} \equiv \rightarrow \equiv \mathscr{S}_1$ be a reduction that consumes messages $J\sigma$ in $\mathscr{A}$, and assume $\mathscr{A} \equiv \mathsf{def}_{S'}\ D\ \mathsf{in}\ \mathscr{A}' \mid J\sigma$. We have the reductions

$$\mathscr{S} \equiv \rightarrow \equiv \mathscr{D}, D, J \triangleright P \vdash_{S \uplus S'} \mathscr{A}' \mid P\sigma \equiv \mathscr{S}_1,$$

$$\mathscr{T} \equiv \rightarrow \equiv \mathscr{D}, D, J \triangleright Q \vdash_{S \uplus S'} \mathscr{A}' \mid Q\sigma \stackrel{\text{def}}{=} \mathscr{T}_1.$$

By definition of ϕ we have $P \approx Q$, by Lemma 14 $P\sigma \approx Q\sigma$, and thus by Lemma 13 $\mathscr{T}_1 \approx (\mathscr{D}, D, J \triangleright Q \vdash_{S \uplus S'} \mathscr{A}' \mid P\sigma)$. Composing these relations, we have $\mathscr{S}_1 \equiv \;\phi\; \approx \mathscr{T}_1$.

We obtain the congruence theorem by structural induction on every context of the open join-calculus, using the above closure property or Lemma 13 at each step.

Proof of Proposition 18. We prove that the relation

$$\phi \stackrel{\text{def}}{=} \approx_{\text{a}} \cup \{(\mathscr{S} \mid x\langle\tilde{v}\rangle, \mathscr{T} \mid x\langle\tilde{v}\rangle) \mid \mathscr{S} \approx_{\text{a}} \mathscr{T}\}$$

is an asynchronous bisimulation up to structural equivalence. We consider only intrusions, all other cases being immediate. Assume $\mathscr{S} \approx_{\text{a}} \mathscr{T}$ and $\mathscr{S} \mid x\langle\tilde{v}\rangle \equiv \stackrel{M}{\longrightarrow}_{\text{J}} \equiv \mathscr{S}'$. By Proposition 15, one of the following holds:

(a) $\mathscr{S} \equiv \stackrel{M}{\longrightarrow}_{\text{J}} \equiv \mathscr{S}''$ with $\mathscr{S}' \equiv \mathscr{S}'' \mid x\langle\tilde{v}\rangle$.
Since $\mathscr{S} \approx_{\text{a}} \mathscr{T}$, there is a solution $\mathscr{T}''$ such that $\mathscr{T} \mid M \Rightarrow \mathscr{T}''$ and $\mathscr{S}'' \approx_{\text{a}} \mathscr{T}''$. The same steps apply in a solution that contains the additional message $x\langle\tilde{v}\rangle$, so $(\mathscr{T} \mid M) \mid x\langle\tilde{v}\rangle \Rightarrow \mathscr{T}'' \mid x\langle\tilde{v}\rangle$ with $\mathscr{S}'' \mid x\langle\tilde{v}\rangle \ \phi \ \mathscr{T}'' \mid x\langle\tilde{v}\rangle$, and thus $\mathscr{S}' \equiv \ \phi \ \mathscr{T}'' \mid x\langle\tilde{v}\rangle$.

(b) $\mathscr{S} \equiv \stackrel{M \mid x\langle\tilde{v}\rangle}{\longrightarrow}_{\text{J}} \equiv \mathscr{S}'$. Since $\mathscr{S} \approx_{\text{a}} \mathscr{T}$, there is a solution $\mathscr{T}''$ such that $\mathscr{T} \mid (M \mid x\langle\tilde{v}\rangle) \Rightarrow \mathscr{T}'$ and $\mathscr{S}' \ \phi \ \mathscr{T}'$, as required. □

Proof of Proposition 20. We prove $\approx_{\text{g}} = \approx$. (We obtain $\approx_{\text{g}} = \approx_{\text{ag}}$ with the same proof of as for Theorem 17.) By definition, $\approx \subseteq \approx_{\text{g}}$. To establish the converse inclusion, we let

$$\phi \stackrel{\text{def}}{=} \{(\mathscr{S}\sigma, \mathscr{T}\sigma) \mid \mathscr{S} \approx_{\text{g}} \mathscr{T} \text{ and } \sigma \text{ global renaming for } \mathscr{S} \text{ and } \mathscr{T}\}$$

and prove that ϕ is a weak bisimulation up to structural equivalence. Let $\mathscr{S}\sigma \ \phi \ \mathscr{T}\sigma$. The only problematic transition is an intrusion $\mathscr{S}\sigma \equiv \stackrel{x\langle\tilde{w}\rangle}{\longrightarrow} \equiv \mathscr{S}'$ when $\tilde{w}$ contains free and/or extruded names. By Lemma 7(3), $\mathscr{S} \equiv \stackrel{y\langle\tilde{v}\rangle}{\longrightarrow} \equiv \mathscr{S}''$ with $\mathscr{S}' \equiv \mathscr{S}''\sigma$ and $x\langle\tilde{w}\rangle = (y\langle\tilde{v}\rangle)\sigma$. Names in $\tilde{v}$ may contain free and/or extruded names. Nonetheless, the label $y\langle\tilde{v}\rangle$ can be written $y\langle\tilde{u}\rho\rangle$ for some distinct fresh names u_i, and $\mathscr{S}$ can perform a ground intrusion $\mathscr{S} \stackrel{y\langle\tilde{u}\rangle}{\longrightarrow} \mathscr{S}_1$ such that $S_1\rho \equiv \mathscr{S}''$ and thus $S_1\rho\sigma \equiv \mathscr{S}'$.

The ground bisimulation clause now applies and yields transitions from $\mathscr{T}$. By Lemma 7(1,3), these transitions carry over to $\mathscr{T}\rho\sigma$:

$$\mathscr{T} \Rightarrow \stackrel{y\langle\tilde{u}\rangle}{\longrightarrow} \Rightarrow \mathscr{T}',$$
$$\mathscr{T}\rho\sigma \Rightarrow \stackrel{x\langle\tilde{w}\rangle}{\longrightarrow} \Rightarrow \mathscr{T}'\rho\sigma$$

with $\mathscr{S}_1 \approx_{\text{g}} \mathscr{T}'$. Since $\mathscr{T}\rho = \mathscr{T}$, the latter series of transition show that $\mathscr{S}_1(\rho\sigma) \ \phi \ \mathscr{T}'(\rho\sigma)$. □

Acknowledgements

Preliminary results in collaboration with Michele Boreale appeared in [7]. We thank the anonymous referees for their constructive remarks. We also thank Jean-Jacques

Lévy, Massimo Merro, Uwe Nestmann, and especially Georges Gonthier for their fruitful comments.

References

[1] M. Abadi, C. Fournet, G. Gonthier, Secure implementation of channel abstractions. Proc. 13th Symp. on Logic in Computer Science. (LICS'98), IEEE Press, New York, June 1998, pp. 105–116.
[2] M. Abadi, C. Fournet, G. Gonthier, Authentication primitives and their compilation, Proc. 27th ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages (POPL'00) ACM Press, New York, January 2000, pp. 302–315.
[3] R.M. Amadio, in: An asynchronous model of locality, failure, and process mobility, Proc. COORDINATION'97, Lecture Notes in Computer Science, Vol. 1282, Springer, Berlin, 1997.
[4] R.M. Amadio, I. Castellani, D. Sangiorgi, On bisimulations for the asynchronous π-calculus, Theoret. Comput. Sci. 195 (2) (1998) 291–324.
[5] J.-P. Banâtre, D.L. Métayer, The Gamma model and its discipline of programming, Sci. Comput. Programming 15 (1990) 55–77.
[6] M. Boreale, On the expressiveness of internal mobility in name-passing calculi, Theoret. Comput. Sci. 195 (2) (1998) 205–226.
[7] M. Boreale, C. Fournet, C. Laneve, Bisimulations in the join-calculus, Proc. PROCOMET'98, IFIP, Chapman & Hall, London, June 1998, pp. 68–86.
[8] M. Boreale, R.D. Nicola, R. Pugliese, in: Proof techniques for cryptographic processes, Proc. 14th Symp. on Logic in Computer Science (LICS'99), IEEE Press, New York, July 1999.
[9] M. Boreale, D. Sangiorgi, Bisimulation in name-passing calculi without matching, Proc. LICS'98, IEEE Press, New York, June 1998.
[10] M. Boreale, D. Sangiorgi, Some congruence properties for π-calculus bisimilarities, Theoret. Comput. Sci. 198 (1–2) (1998) 159–176.
[11] G. Boudol, Asynchrony and the π-calculus (note), Rapport de recherche 1702, INRIA Sophia-Antipolis, May 1992.
[12] S. Conchon, F. Le Fessant, Jocaml: mobile agents for objective-caml, Proc. ASA/MA'99 IEEE Press, October, New York, 1999, pp. 22–29.
[13] P. Degano, R. Gorrieri, A. Marchetti-Spaccamela (Eds.), in: Proc. 24th Colloq. on Automata, Languages and Programming (ICALP'97), Lecture Notes in Computer Science, Vol. 1256, Springer, Berlin, July 1997.
[14] C. Fournet, The join-calculus: a calculus for distributed mobile programming, Ph.D. Thesis, Ecole Polytechnique, Palaiseau, November 1998.
[15] C. Fournet, G. Gonthier, The reflexive chemical abstract machine and the join-calculus, Proc. POPL'96, ACM Press, New York, January 1996, pp. 372–385.
[16] C. Fournet, G. Gonthier, A hierarchy of equivalences for asynchronous calculi (extended abstract), In: Larsen et al. [24], pp. 844–855.
[17] C. Fournet, G. Gonthier, J.-J. Lévy, L. Maranget, D. Rémy, A calculus of mobile agents, in: U. Montanari, V. Sassone (Eds.), Proc. 7th Internat. Conf. on Concurrency Theory (CONCUR'96), Lecture Notes in Computer Science, Vol. 1119, Springer, Berlin, August 1996,, pp. 406–421.
[18] C. Fournet, C. Laneve, L. Maranget, D. Rémy, Implicit typing à la ML for the join-calculus, in: A. Mazurkiewicz, J. Winkowski (Eds.) (Eds.), Proc. 8th Internat. Conf. Concurrency Theory, Lecture Notes in Computer Science, Vol. 1243, Springer, Berlin, July 1997, pp. 196–212.
[19] C. Fournet, L. Maranget, The join-calculus language (version 1.03 beta), Source distribution and documentation available from `http://join.inria.fr/`, June 1997.
[20] K. Honda, M. Tokoro, An object calculus for asynchronous communication, in: P. America (Ed.), Proc. ECOOP'91 Conf., Vol. 512 Springer, Berlin, 1991, 133–147.
[21] K. Honda, M. Tokoro, On asynchronous communication semantics, in: P. Wegner, M. Tokoro, O. Nierstrasz (Eds.), Proc. of the ECOOP'91 Workshop on Object-Based Concurrent Computing, Lecture Notes in Computer Science, Vol. 612, Springer, Berlin, 1992, pp. 21–51.
[22] K. Honda, N. Yoshida, On reduction-based process semantics, Theoret. Comput. Sci. 152 (2) (1995) 437–486.

[23] C. Laneve, May and must testing in the join-calculus. Technical Report UBLCS 96-04, University of Bologna, March 1996 (Revised: May 1996).
[24] K. Larsen, S. Skyum, G. Winskel (Eds.), Proc. 25th Internat. Colloq. on Automata, Languages and Programming (ICALP'98), Lecture Notes in Computer Science, Vol. 1443, Springer, Berlin, 1998.
[25] M. Merro, D. Sangiorgi, On asynchrony in name-passing calculi, in: K. Larsen, S. Skyum, G. Winskel, (Eds.), [24], pp. 856–867.
[26] R. Milner, Communication and Concurrency, Prentice Hall, New York, 1989.
[27] R. Milner, The polyadic π-calculus: a tutorial, in: F.L. Bauer, W. Brauer, H. Schwichtenberg (Eds.), Logic and Algebra of Specification, Springer, Berlin, 1993.
[28] R. Milner, J. Parrow, D. Walker, A calculus of mobile processes Parts I and II. Inform. and Comput., 100 1992 (1–40 and 41–77).
[29] R. Milner, D. Sangiorgi, Barbed bisimulation, in: W. Kuich (Ed.), Proc. ICALP'92, Lecture Notes in Computer Science, Vol. 623, Springer, Berlin, 1992, pp. 685–695.
[30] U. Montanari, M. Pistore, Checking bisimilarity for finitary π-calculus, in: I. Lee, S.A. Smolka (Eds.), Proc. 6th Internat. Conf. on Concurrency Theory (CONCUR'95), Lecture Notes in Computer Science, Vol. 962, Springer, Berlin, 1995, pp. 42–56.
[31] U. Nestmann, On the expressive power of joint input, in: C. Palamidessi, I. Castellani (Eds.), EXPRESS'98: expressiveness in Concurrency, Electronic Notes in Theoretical Computer Science, Vol. 16.2, Elsevier, Amsterdam, September 1998.
[32] J. Riely, M. Hennessy, Distributed processes and location failures, in: P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela (Eds.), [13], pp. 471–481.
[33] D. Sangiorgi, Expressing mobility in process algebras: first-order and higher-order paradigms, Ph.D. Thesis, University of Edinburgh, May 1993.
[34] D. Sangiorgi, On the bisimulation proof method, Revised version of Technical Report ECS-LFCS-94-299, University of Edinburgh, 1994, An extended abstract appears in Proc. of MFCS'95, Lecture Notes in Computer Science, Vol. 969, 1994.
[35] D. Sangiorgi, The name discipline of uniform receptiveness, in: P. Degano, R. Gorrieri, A. Marchetti-Spaccamela, (Eds.), [13], pp. 303–313.
[36] D. Sangiorgi, R. Milner, The problem of "weak bisimulation up to", in: W.R. Cleaveland (Ed.), Proc. CONCUR'92, Lecture Notes in Computer Science, Vol. 630, Springer, Berlin, 1992, pp. 32–46.
[37] B. Victor, F. Moller, The mobility workbench – a tool for the π-calculus: in: Proc. CAV'94, Lecture Notes in Computer Science, Vol. 818, Springer, Berlin, 1994, pp. 428–440.